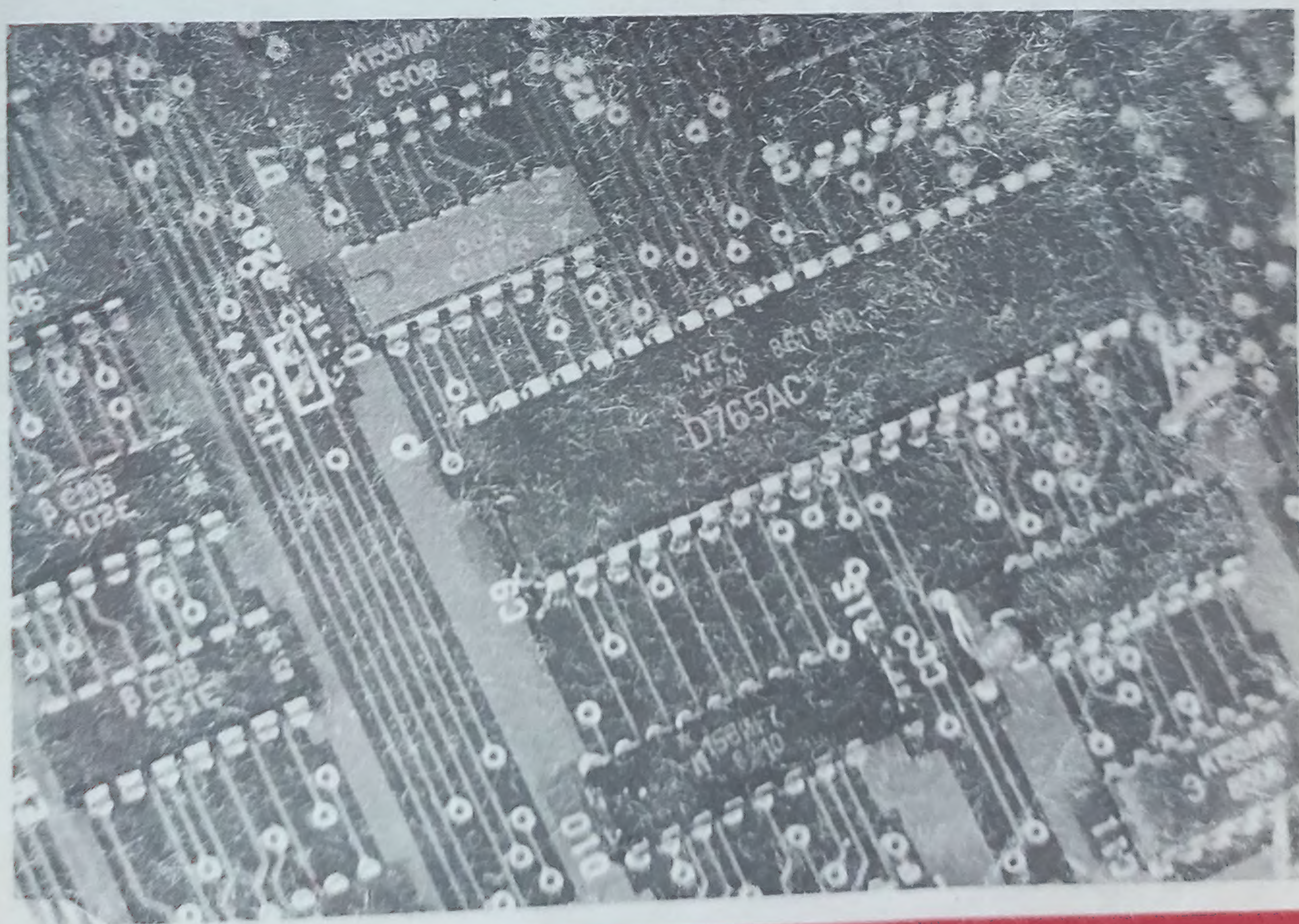


III  
20236

MARIUS CORNEA HAŞEGAN  
DORINA CORNEA HAŞEGAN

# PROIECTAREA SISTEMELOR cu MICROPROCESOR **Z80**



**DACIA**



CORNEA HAŞEGAN MARIUS • CORNEA HAŞEGAN DORINA

# PROIECTAREA SISTEMELOR CU MICROPROCESOR Z 80 APLICAȚII



## CUPRINS

<b>CAPITOLUL 1</b>	<b>Introducere</b>	6
<b>CAPITOLUL 2</b>	<b>Unitatea centrală de prelucrare a sistemului Z80—Z80 CPU</b>	12
2.1.	Structura unității centrale a sistemului Z80	12
2.2.	Descrierea funcțiilor logice ale circuitului Z80 CPU	14
2.3.	Tratarea întreruperilor	16
<b>CAPITOLUL 3</b>	<b>Programarea microprocesorului Z80</b>	18
3.1.	Setul instrucțiunilor executabile ale microprocesorului Z80	18
3.2.	Indicatori de condiții și operații aritmetice	38
3.3.	Exemple de utilizare a instrucțiunilor specifice microprocesorului Z80	43
<b>CAPITOLUL 4.</b>	<b>Circuitul numărător-temporizator Z80 CTC</b>	48
4.1.	Descrierea circuitului numărător-temporizator Z80 CTC	48
4.2.	Aplicații ale circuitului Z80 CTC	56
<b>CAPITOLUL 5.</b>	<b>Circuitul de control pentru intrare-ieșire serială, Z80 SIO</b>	64
5.1.	Descrierea circuitului de control pentru intrare-ieșire serială	64
5.2.	Aplicații ale circuitului Z80 SIO	86
<b>CAPITOLUL 6.</b>	<b>Circuitul de intrare-ieșire paralelă, Z80 PIO</b>	93
6.1.	Descrierea circuitului de intrare-ieșire paralelă Z80 PIO	93
6.2.	Aplicații ale circuitului Z80 PIO	106
<b>CAPITOLUL 7.</b>	<b>Circuitul de control al accesului direct la memorie, Z80 DMA</b>	118
7.1.	Descrierea circuitului Z80 DMA	118
7.2.	Aplicații ale circuitului Z80 DMA	136
<b>CAPITOLUL 8.</b>	<b>Realizarea unui sistem Z80. Aplicații</b>	139
8.1.	Elementele unui sistem cu microprocesor Z80	139
8.2.	Aplicații	157
<b>CAPITOLUL 9.</b>	<b>Sistem de dezvoltare cu microprocesor Z80</b>	178
9.1.	Structura hardware și sistemul de operare	178
9.2.	Interpretor Basic de 12 ko pentru microprocesorul Z80	189
	<i>Bibliografia</i>	213
<b>ANEXA</b>	<i>Circuite integrate utilizate în sistemele cu microprocesor Z80</i>	214



## CAPITOLUL I

### INTRODUCERE

Apărut în anii 1970, microprocesorul este un circuit capabil să efectueze funcțiile aritmetice și de control ale unui calculator. Este un circuit integrat pe scară largă (LSI), conținând mii de tranzistoare și rezistoare pe o suprafață de aproximativ  $5 \text{ mm}^2$ , consumul de putere tipic fiind de 150 mW.

Pentru a forma un sistem de calcul complet, microprocesorul trebuie utilizat în combinație cu alte circuite suport, incluzând și memorii. S-au dezvoltat astfel două clase de sisteme cu microprocesor: microcalculatoare și sisteme de control cu microprocesor, cum sînt, de exemplu, cele destinate unor mașini de spălat sau de cusut automate și care nu necesită în general introducerea unor programe propriu-zise de către utilizator.

Dezvoltarea microprocesoarelor a urmat dezvoltării circuitelor integrate începută după 1960, complexitatea acestora dublîndu-se practic în fiecare an, pînă în prezent. Dacă se menține ritmul actual, se prevede ca după 1990 să existe circuite integrate VLSI conținînd sute de milioane de tranzistoare fiecare. Figura 1.1 prezintă evoluția aproximativă, pînă în prezent și în perspectivă, a complexității circuitelor integrate.

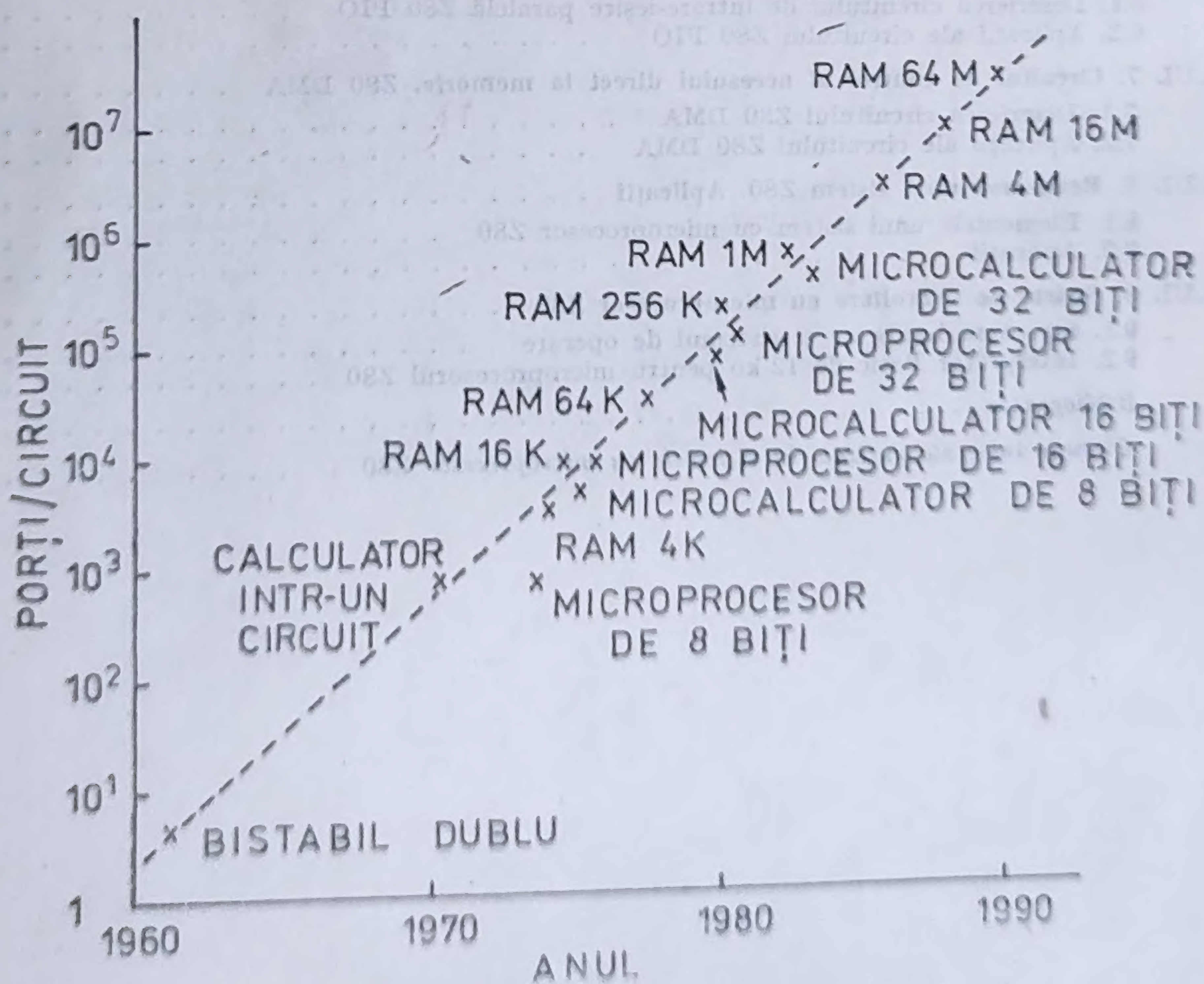


Fig. 1.1 Evoluția complexității circuitelor integrate.



Un precursor al microprocesorului a fost unitatea aritmetico-logică (UAL), apărută după 1960, care putea să adune, să scadă, să rotească și să deplaseze o informație binară, ca și un microprocesor sau un microcalculator în prezent. După 1970, UAL a fost combinată, pe un singur circuit, cu bistabile, registre de deplasare și alte circuite, formînd un microprocesor rudimentar, apărut atunci cînd tehnologia a permis plasarea a 1000 de tranzistoare pe aceeași pastilă.

Cîțiva ani mai tîrziu, în același circuit au fost incluse circuite de intrare/ieșire, memorii volatile (RAM), memorii nevolatile (ROM) și circuite de tact, formînd astfel un microcalculator într-un singur circuit. Linia de demarcație între microprocesoare și microcalculatoare nu este net definită. În general, un microprocesor execută secvențial un șir de instrucțiuni, în timp ce un microcalculator execută instrucțiuni, dar este prevăzut și cu circuite de intrare/ieșire seriale și/sau paralele, accesibile utilizatorului, cu memorii ROM, nevolatile, eventual programabile, PROM, în care utilizatorul își poate depozita programul, cu memorii RAM pentru depozitarea temporară a programelor sau datelor și cu circuite de generare a semnalului de tact, necesitînd în exterior doar un cristal de cuarț sau o rețea RC pentru producerea oscilațiilor.

Un circuit microcalculator este denumit adesea „calculator într-un singur circuit”. Denumirea este improprie, deoarece dispozitivele voluminoase cum ar fi tastatura, dispozitivul de afișare și sursa de putere trebuie adăugate în exterior pentru a completa sistemul. Cele mai multe sisteme cu microprocesor necesită zeci de circuite pentru a le conecta la un proces extern. O limitare importantă în calea dezvoltării posibilităților unui singur circuit o constituie în prezent numărul de conexiuni externe care pot fi create pentru o singură pastilă.

Figura 1.2 ilustrează poziția microprocesorului într-un microcalculator. Unele microcalculatoare într-un singur circuit conțin mai puține elemente decît în figură,

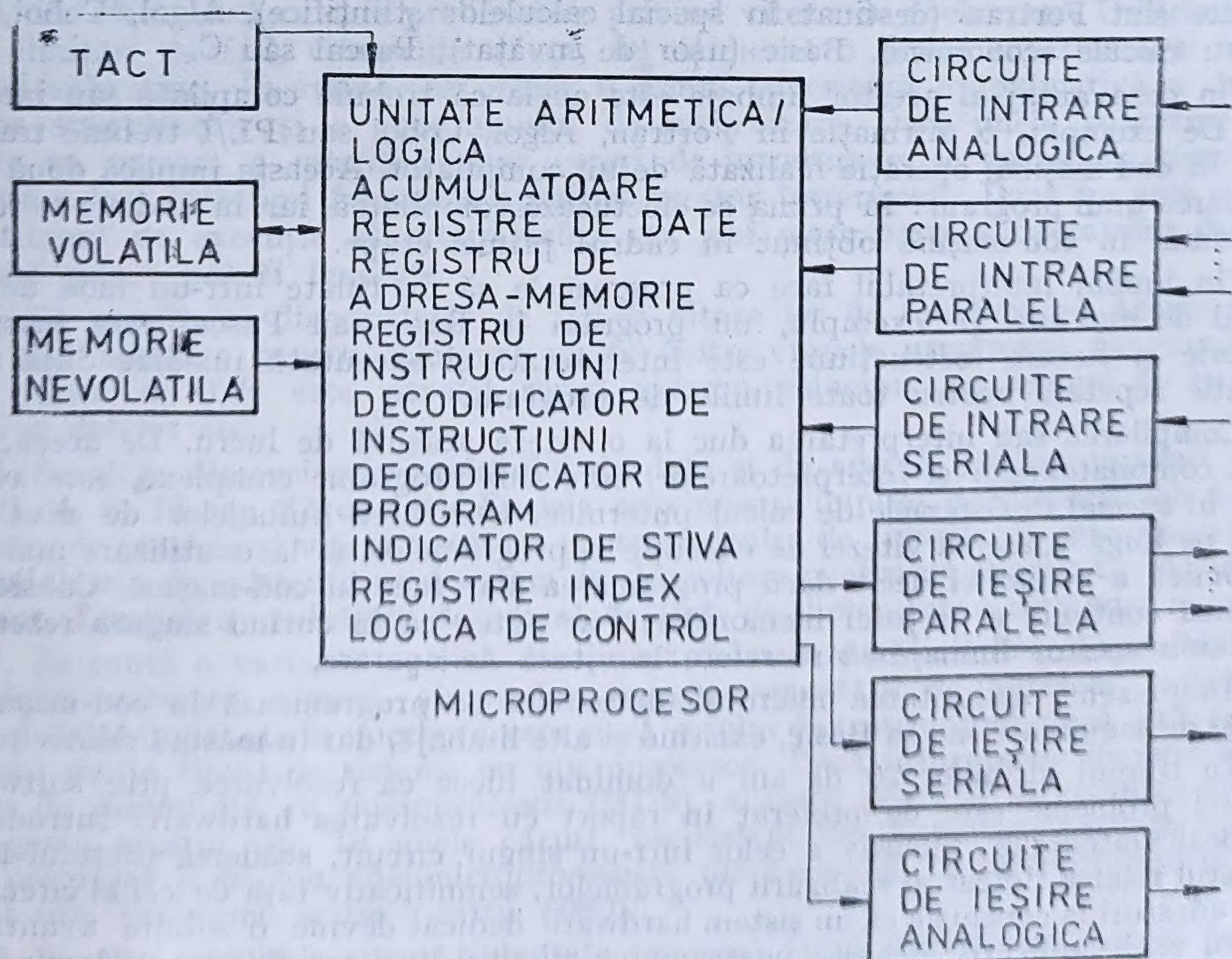


Fig. 1.2 Diagrama bloc a unui microcalculator.



alte conțin mai multe. O serie de microcalculatoare cu destinație specială conțin interfețe de intrare/ieșire aplicabile numai unei mici clase de aplicații.

În paralel cu dezvoltarea calculatoarelor electronice, au apărut termenii hardware, software și firmware.

Termenul hardware desemnează circuitele electronice, sursele de alimentare, interconexiunile cu claviaturi, cititoare/perforatoare de bandă de hîrtie, unități de casetă sau de disc flexibil, dispozitive de afișare, etc.

Crearea părții de hardware revine inginerului proiectant al calculatorului și interesează doar în măsură redusă utilizatorul.

Software se referă la programele care sînt dezvoltate pentru calculator sau microprocesor. Ele pot fi listate pe hîrtie sau codificate pe cartele, bandă de hîrtie, bandă magnetică, disc, sau disc flexibil. Programele trebuie încărcate în calculator înainte de a fi rulate, sarcină care revine programatorului.

La nivelul microcalculatorului (calculatorului), instrucțiunile unui program sînt formate din cifre binare (1 și 0), formînd un program în cod-mașină. În practică, fiecare grup de 4 cifre binare se înlocuiește printr-o cifră hexazecimală (uneori grupuri de 3 cifre binare se înlocuiesc printr-o cifră octală). Codurilor binare sau hexazecimale li se asociază, pentru a putea fi memorate, grupuri de litere, numite mnemonice, sugerînd operația efectuată. Rezultă astfel limbajul de asamblare asociat microcalculatorului. Un program special, numit asamblor, poate transforma un șir de mnemonice în cod-mașină. Transformarea inversă, din cod-mașină în limbaj de asamblare este realizată de programe numite dezasambleare.

O treaptă superioară în programare o reprezintă limbajele de nivel înalt, apropiate de vorbirea obișnuită. Totuși, regulile gramaticii fiecărui limbaj trebuie respectate cu strictețe pentru ca microcalculatorul, ca și orice calculator în general, să poată executa instrucțiunile programatorului. Dintre limbajele de nivel înalt, cele mai utilizate sînt Fortran (destinat în special calculului științific), Algol, Cobol, PL/I (pentru calcule economice), Basic (ușor de învățat), Pascal sau C.

Un dezavantaj al acestor limbaje este acela că trebuie compilate sau interpretate. De exemplu, o afirmație în Fortran, Algol, Cobol sau PL/I trebuie transformată în cod-mașină, operație realizată de un compilator. Aceasta implică două etape în rularea unui program: în prima se efectuează compilarea, iar în a doua se rulează programul în cod-mașină obținut în cadrul primei etape.

Un limbaj interpretabil face ca programele să fie rulate într-un mod deosebit de cel de mai sus. De exemplu, un program în Basic sau Pascal este înscris în memorie și fiecare instrucțiune este interpretată și executată imediat după aceea, operație repetată pentru toate liniile de program.

Compilarea sau interpretarea duc la o viteză scăzută de lucru. De aceea, utilizarea compilatoarelor și interpretoarelor, care sînt programe complexe, este avantajoasă în special în sistemele de calcul puternice. Utilizarea limbajelor de nivel înalt duce, pe lîngă scăderea vitezei de execuție a programelor, și la o utilizare mai puțin economică a memoriei decît dacă programarea s-ar face în cod-mașină. Considerînd scăderea continuă a prețului memoriilor, s-ar putea ca în curînd singura rezervă în utilizarea acestor limbaje să se refere la viteza de operare.

În prezent, majoritatea microcalculatoarelor se programează în cod-mașină, în limbaj de asamblare sau în Basic, existînd și alte limbaje, dar în măsură relativ redusă.

În timpul ultimilor 20 de ani a dominat ideea că rezolvarea prin software a oricărei probleme este de preferat în raport cu rezolvarea hardware. Introducerea microcalculatoarelor, inclusiv a celor într-un singur circuit, scăderea prețului lor, ca și costul relativ ridicat al realizării programelor, semnificativ față de cel al circuitelor, duce adeseori la concluzia că un sistem hardware dedicat devine o soluție avantajoasă pentru unele aplicații. Aceasta poate marca sfîrșitul unei ere în care a dominat calculatorul de mari dimensiuni, deservind un mare număr de utilizatori.



alte conțin mai multe. O serie de microcalculatoare cu destinație specială conțin interfețe de intrare/ieșire aplicabile numai unei mici clase de aplicații.

În paralel cu dezvoltarea calculatoarelor electronice, au apărut termenii hardware, software și firmware.

Termenul hardware desemnează circuitele electronice, sursele de alimentare, interconexiunile cu claviaturi, cititoare/perforatoare de bandă de hârtie, unități de casetă sau de disc flexibil, dispozitive de afișare, etc.

Crearea părții de hardware revine inginerului proiectant al calculatorului și interesează doar în măsură redusă utilizatorul.

Software se referă la programele care sînt dezvoltate pentru calculator sau microprocesor. Ele pot fi listate pe hârtie sau codificate pe cartele, bandă de hârtie, bandă magnetică, disc, sau disc flexibil. Programele trebuie încărcate în calculator înainte de a fi rulate, sarcină care revine programatorului.

La nivelul microcalculatorului (calculatorului), instrucțiunile unui program sînt formate din cifre binare (1 și 0), formînd un program în cod-mașină. În practică, fiecare grup de 4 cifre binare se înlocuiește printr-o cifră hexazecimală (uneori grupuri de 3 cifre binare se înlocuiesc printr-o cifră octală). Codurilor binare sau hexazecimale li se asociază, pentru a putea fi memorate, grupuri de litere, numite mnemonice, sugerînd operația efectuată. Rezultă astfel limbajul de asamblare asociat microcalculatorului. Un program special, numit asamblor, poate transforma un șir de mnemonice în cod-mașină. Transformarea inversă, din cod-mașină în limbaj de asamblare este realizată de programe numite dezasambleare.

O treaptă superioară în programare o reprezintă limbajele de nivel înalt, apropiate de vorbirea obișnuită. Totuși, regulile gramaticii fiecărui limbaj trebuie respectate cu strictețe pentru ca microcalculatorul, ca și orice calculator în general, să poată executa instrucțiunile programatorului. Dintre limbajele de nivel înalt, cele mai utilizate sînt Fortran (destinat în special calculelor științifice), Algol, Cobol, PL/I (pentru calcule economice), Basic (ușor de învățat), Pascal sau C.

Un dezavantaj al acestor limbaje este acela că trebuie compilate sau interpretate. De exemplu, o afirmație în Fortran, Algol, Cobol sau PL/I trebuie transformată în cod-mașină, operație realizată de un compilator. Aceasta implică două etape în rularea unui program: în prima se efectuează compilarea, iar în a doua se rulează programul în cod-mașină obținut în cadrul primei etape.

Un limbaj interpretabil face ca programele să fie rulate într-un mod deosebit de cel de mai sus. De exemplu, un program în Basic sau Pascal este înscris în memorie și fiecare instrucțiune este interpretată și executată imediat după aceea, operație repetată pentru toate liniile de program.

Compilarea sau interpretarea duc la o viteză scăzută de lucru. De aceea, utilizarea compilatoarelor și interpretoarelor, care sînt programe complexe, este avantajoasă în special în sistemele de calcul puternice. Utilizarea limbajelor de nivel înalt duce, pe lîngă scăderea vitezei de execuție a programelor, și la o utilizare mai puțin economică a memoriei decît dacă programarea s-ar face în cod-mașină. Considerînd scăderea continuă a prețului memoriilor, s-ar putea ca în curînd singura rezervă în utilizarea acestor limbaje să se refere la viteza de operare.

În prezent, majoritatea microcalculatoarelor se programează în cod-mașină, în limbaj de asamblare sau în Basic, existînd și alte limbaje, dar în măsură relativ redusă.

În timpul ultimilor 20 de ani a dominat ideea că rezolvarea prin software a oricărei probleme este de preferat în raport cu rezolvarea hardware. Introducerea microcalculatoarelor, inclusiv a celor într-un singur circuit, scăderea prețului lor, ca și costul relativ ridicat al realizării programelor, semnificativ față de cel al circuitelor, duce adeseori la concluzia că un sistem hardware dedicat devine o soluție avantajoasă pentru unele aplicații. Aceasta poate marca sfîrșitul unei ere în care a dominat calculatorul de mari dimensiuni, deservind un mare număr de utilizatori.



Termenul firmware, apărut odată cu dezvoltarea microprocesoarelor, desemnează programele înscrise permanent în memoria sistemelor utilizând aceste circuite, și care sînt gata de utilizare la punerea sub tensiune, fără a fi necesară introducerea lor de către utilizator. Ca exemple, pot fi menționate programele de aplicații înscrise în memoria sistemelor care deservesc mașini de uz casnic sau industrial, ca și programele rezidente în microcalculatoarele denumite personale, cum ar fi sistemele de operare sau interpretoarele Basic.

Aplicațiile implicînd microprocesoare și microcalculatoare vor afecta în următorii ani toate aspectele activității umane. Proiectarea acestora nu se poate face după un șablon unic. Totuși, o abordare secvențială a problemei este întotdeauna posibilă, într-o ordine care, cel puțin parțial, poate respecta etapele de lucru prezentate în continuare:

1. Se definesc cerințele și obiectivele proiectului. Această etapă este utilă nu numai proiectantului, dar și altor persoane care pot face observații și pot sugera îmbunătățiri.

2. Se trasează o diagramă bloc simplificată a structurii hardware, utilizînd informațiile de la pasul 1. Se observă dacă sistemul de cerințe și obiective definit anterior este complet sau dacă trebuie modificat sau extins. Se poate face o primă apreciere asupra costului proiectului și se pot preciza primele componente necesare.

3. Se trasează o schemă logică simplificată care ilustrează relația dintre programul de supervizare (programul principal) și subrutinele mai importante. Se asignează subrutine specifice pentru controlul intrărilor și ieșirilor. Un minim pentru aceste funcții specifice de control trebuie furnizat de către programul de supervizare, care trebuie să servească doar ca o legătură între diferitele subrutine. Toată „munca” trebuie efectuată de către acestea.

4. Se efectuează calcule preliminare privind viteza de prelucrare. Utilizînd diferitele limitări de I/E, cunoscute acum, se estimează o limită superioară a duratei fiecărei subrutine. Însumînd, se obține o limită superioară aproximativă a duratei tuturor subrutinelor și a programului de supervizare. Dacă unele subrutine sînt apelate ca urmare a apariției unor cereri de întrerupere, se va estima timpul de execuție a lor, apreciind frecvența apariției acestor întreruperi. Dacă nu este important timpul de execuție al programelor, cea mai mare parte a operațiilor descrise la acest punct pot fi ignorate.

5. Se precizează dispozitivele de I/E și viteza lor de funcționare. Aceste informații sînt necesare pentru cele mai multe dintre etapele următoare. Se precizează dacă portul de I/E este paralel, serial, sincron, asincron, cu semnal de tact, cu captarea datelor etc.

6. Se alege dimensiunea cuvîntului de date și de adresă. Microprocesorul poate fi de 1, 4, 8, 16 sau 32 de biți. Decizia este uneori impusă de dispozitivele de I/E utilizate, de precizia și viteza necesare pentru calcule, de tipurile de microprocesoare disponibile sau de existența unui sistem de dezvoltare pentru un anumit tip de microprocesor. Lungimea cuvîntului de adresă depinde de dimensiunea memoriei necesare.

7. Se caută o variantă optimă din punct de vedere al costului, disponibilității, suportului software, vitezei, circuitelor suport, capacității de adresare, capacității RAM și ROM înglobate în microprocesor și al setului de instrucțiuni, dacă sînt accesibile mai multe tipuri de sisteme cu microprocesor. Dacă utilizatorul dispune de un sistem de dezvoltare cu microprocesor (MDS), aceasta poate fi un avantaj decisiv în alegerea acestui tip. În unele cazuri, un sistem de dezvoltare permite rularea unor programe și pentru alte microprocesoare (de exemplu, cu microprocesorul Z80 se pot rula programe scrise pentru 8080).

8. Se aleg circuitele suport integrate pe scară largă (LSI), convenabile pentru microprocesorul selectat. Trebuie asigurată o documentație amănunțită pentru aceste circuite. Se proiectează circuitul generator de tact, amplificatoarele de magistrală



și circuitele principale pentru decodificarea adreselor. Acestea formează „inima” sistemului și trebuie să aibă capacitatea de a controla întregul sistem, cu rezerve pentru eventuale dezvoltări ulterioare.

9. Se proiectează memoria sistemului, utilizând circuite RAM, ROM, PROM, EPROM, EAROM etc. Dacă este posibil, se alege un microprocesor cu cât mai multă memorie înglobată. Dacă se prevede o producție în serie pentru dispozitivul proiectat, se aleg și memorii PROM pentru dezvoltare și memorii ROM, programabile prin mască, pentru producția industrială.

10. Se proiectează circuitele de I/E, utilizând circuite LSI pentru care se dispune de documentație adecvată. Se mențin magistralele principale de date și de adrese ale microprocesorului la o distanță de maximum câțiva centimetri de microprocesor, pentru a evita apariția perturbațiilor datorate paraziților. Pentru interfațarea cu alte plăci sau dispozitive de I/E, se utilizează magistrale de date secundare și linii de adresă decodificate. Se evită multe probleme de dezvoltare menținând magistralele primare, având în general un nivel redus de putere, pe o singură placă și într-o zonă strict controlată.

11. Se proiectează panoul de control și alte interfețe pentru utilizatorul uman. Se prevăd cât mai multe posibilități de depanare, în limitele admise de costul proiectului. Realizând panoul de control astfel încât să funcționeze sub supravegherea programului (atât intrările cât și mărimile vizualizate), se pot încorpora în partea de hardware și software instrumente utile care pot economisi sute de ore de depanare a programelor noi. Aceste instrumente pot fi extrem de utile și în timpul dezvoltării sistemului sau pentru identificarea unor erori.

12. Se proiectează sistemul de alimentare. Sursele de alimentare se pot proiecta sau achiziționa. Trebuie avută în vedere o rezervă de putere pentru fiecare tensiune de alimentare, pentru eventualitatea dezvoltării sistemului. Se proiectează circuitele imprimate, cablajul și conectorii sistemului. Se prevăd capacități de deparazitare în număr suficient pe fiecare placă pentru a menține oscilațiile tensiunilor de alimentare în domeniul milivolților.

13. Fiind cunoscute mai multe elemente despre sistem, se trasează o schemă logică detaliată a programului de supervizare. Se stabilește localizarea, mărimea, tipul și informațiile de intrare/ieșire pentru diferitele subrutine. Se recomandă utilizarea programării modulare și se mențin, în limita posibilităților, rutinele independente unele în raport cu altele. Subrutinele considerate optime au între 20 și 50 de instrucțiuni, ceea ce duce la simplificarea depanării sau corectării lor, atât pentru cel care le scrie cât și pentru alți utilizatori care doresc să intervină în program.

14. Se formează o hartă a memoriei, cu toate informațiile necesare. Este util să se plaseze aceste informații înaintea listării programului, creînd posibilitatea de a fi reactualizate împreună cu programul.

15. Se scrie programul de supervizare și toate subrutinele. Se assemblează, se depanează, se reassemblează și se pregătește programul pentru testare. Dacă este posibil, se simulează partea hardware cu ajutorul sistemului de dezvoltare cu microprocesor (MDS) și se încearcă rularea programului.

16. Se încarcă programul asamblat în RAM, PROM sau într-un simulator de ROM și se testează pe sistemul proiectat. Se depanează, se reprojectează și se rescrie programul dacă este cazul, pentru a aduce sistemul în stare de funcționare. Această etapă necesită cel mai lung interval de timp și este cea în care proiectele supraviețuiesc sau trebuie să se renunțe la ele.

17. Înaintea listării programului, se listează toți parametrii de proiectare, registrele temporare, registrele de lucru din memorie (scratch-pad memory), porturile de I/E etc, descriind fiecare element în întregime.

Sistemul cu microprocesor Z80, unul dintre cele mai sofisticate, versatile și utilizate dintre sistemele cu microprocesor de 8 biți produse până în prezent, permite,



în mare, urmărirea etapelor descrise mai sus în activitatea de proiectare și de execuție. Existența unor sisteme cu microprocesor Z80 produse industrial și adecvate multor aplicații simplifică această activitate. Totuși, și în continuare o serie de proiecte vor necesita sisteme cu microprocesor dedicate unor aplicații particulare și care vor trebui realizate într-un mod asemănător celui prezentat.

Capitolele următoare ale prezentului volum descriu, pe rând, componentele familiei Z80, realizarea și câteva aplicații ale unui sistem Z80, ca și un sistem de operare și un interpretor Basic, destinate sistemului descris.



## CAPITOLUL II

### UNITATEA CENTRALĂ DE PRELUCRARE A SISTEMULUI Z 80— Z 80 CPU

Unitatea centrală de prelucrare este elementul în jurul căruia se formează sistemul cu microprocesor. Rolul ei este de a aduce instrucțiuni din memorie și de a realiza operațiile dorite. Familia dispozitivelor de intrare/ieșire și a memoriilor direct compatibile cu Z 80 CPU este suficient de variată pentru a realiza sisteme într-o gamă largă, practic fără circuite logice suplimentare. Activitatea de realizare a programelor devine astfel preponderentă față de cea de dezvoltare a suportului hardware, în majoritatea aplicațiilor.

Setul de instrucțiuni conține 158 de instrucțiuni dintre care 78 sînt ale microprocesorului 8080. Instrucțiunile permit prelucrarea unor șiruri de date de 1 sau 2 octeți sau a unor biți. Instrucțiunile de transfer de bloc sau de căutare într-un bloc de date și de asemenea modurile de adresare indexată sau relativă, fac ca posibilitățile de prelucrare a datelor să situeze microprocesorul Z 80 pe primul loc în categoria microprocesoarelor de 8 biți.

Principalele caracteristici ale acestui microprocesor sînt: — existența unui sistem de întreruperi, orientat pentru componentele sistemului; set dublu de registre; posibilitatea de a trata întreruperile în trei moduri diferite; existența unui numărător înglobat în microprocesor, pentru reîncărcarea memoriilor RAM dinamice; existența mai multor variante de unitate centrală (Z 80 CPU — 2,5 MHz, Z 80 A — 4 MHz, Z 80 B — 6 MHz, Z 8400 L — 1 — 1 MHz/15 mA, Z 8400 L — 2 — 1,5 MHz/20 mA, Z 8400 L — 3 — 2,5 MHz/25 mA, ultimele trei fiind produse de firma MOSTEK); set de instrucțiuni care conferă microprocesorului cea mai mare capacitate de prelucrare între microprocesoarele de 8 biți din generația a treia; 208 biți de memorie RAM accesibili în memoria internă; o singură tensiune de alimentare de +5 V; semnalele de ieșire permit conectarea directă la memoriile și circuitele periferice obișnuite.

#### 2.1. STRUCTURA UNITĂȚII CENTRALE A SISTEMULUI Z 80

Schema bloc a unității centrale este reprezentată în figura 2.1.

Registrele unității centrale formează trei grupuri distincte.

Primul grup conține două seturi de registre de 8 biți: unul principal și unul secundar notat cu "''". Ambele seturi constau dintr-un acumulator A (A'), un registru al indicatorilor de condiții F(F') și 6 registre de uz general: B, C, D, E, H, L (B' C' D' E' H' L'). Transferul datelor între aceste două seturi se realizează prin utilizarea unor instrucțiuni de schimb, rezultînd posibilitatea unui răspuns mai rapid la cererile de întrerupere, ca și aceea a prelucrării datelor în două planuri, utilizînd setul principal, respectiv cel secundar.

Registrul acumulator păstrează rezultatul unei operații aritmetice sau logice de 8 biți, iar indicatorii de condiții din registrul F oferă informații despre rezultatul unor operații de 8 sau 16 biți, așa cum se prezintă în tabelele 3.3, 3.4, 3.5, 3.6 și 3.7.

Registrele de uz general pot fi utilizate individual, ca registre de 8 biți, sau în perechi, ca registre de 16 biți: BC, DE, HL (BC', DE', HL').



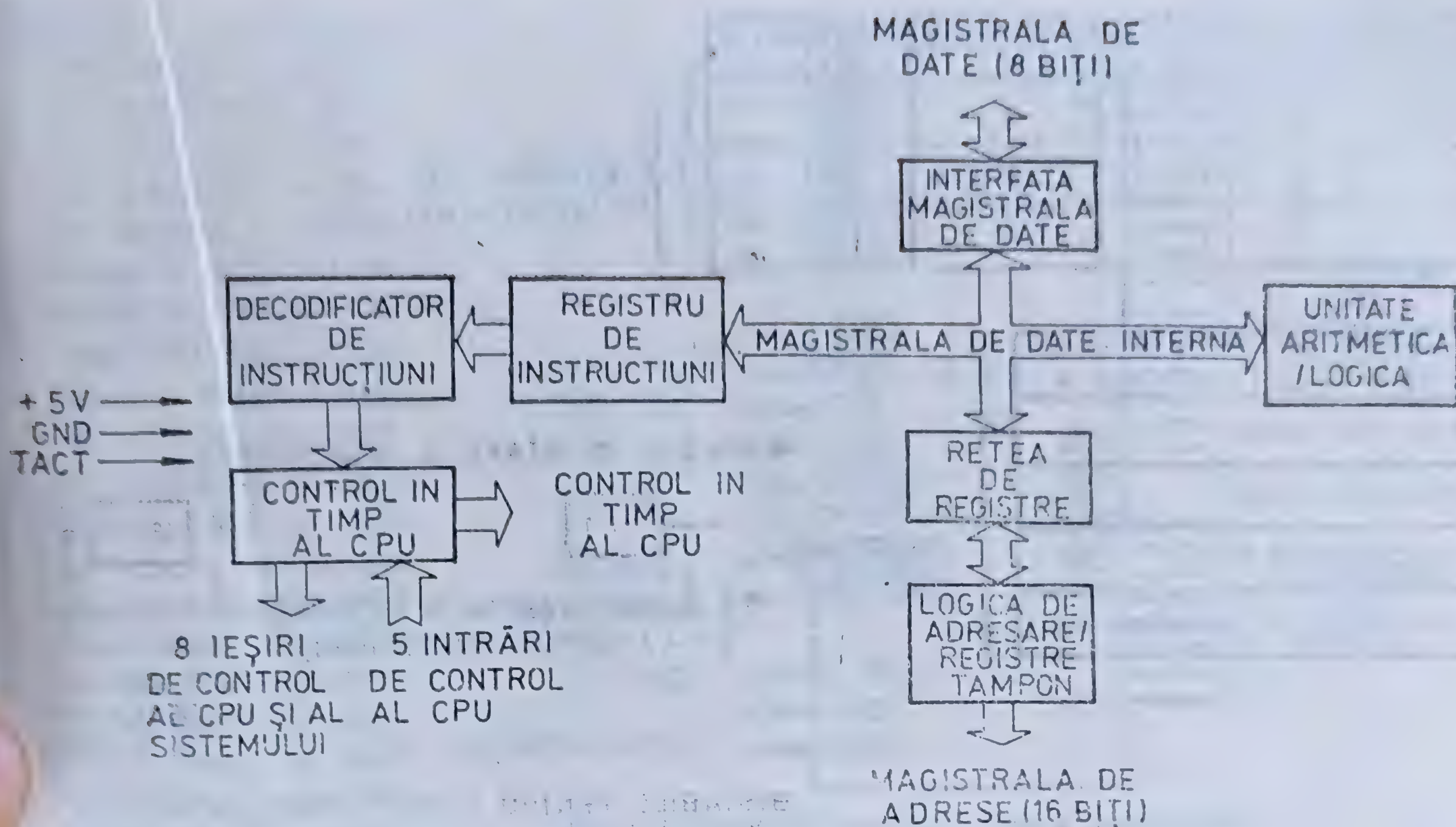


Fig. 2.1 Schema bloc a unității centrale Z80 CPU.

Al doilea grup de registre conține registrele cu destinație specială: numărătorul de program (PC), care conține adresa de 16 biți a instrucțiunii care este adusă din memorie, fiind incrementat imediat după ce conținutul lui a fost transferat pe liniile de adresă, și încărcat cu noua valoare a adresei atunci când în program apare un salt; indicatorul de stivă (SP), care conține adresa de 16 biți a vârfului zonei de memorie RAM numită stivă, organizată după principiul LIFO (ultimul intrat — primul ieșit), servind pentru depunerea unor date din registrele unității centrale pentru preluare ulterioară; registrele index (IX) și (IY), independente, care conțin câte o adresă de bază, de 16 biți, utilizată în modul de adresare indexată, în care conținutul registrului index reprezintă adresa de bază a unei zone de memorie în care se află datele cu care se lucrează; un octet suplimentar este inclus în instrucțiunile cu adresare indexată pentru a specifica deplasamentul față de adresa de bază, sub forma unui întreg, cu semn, în aritmetica complementului față de 2; acest mod de adresare simplifică în special prelucrarea tabelelor de date; registrul de adresă al paginii de întrerupere (I) conține cei 8 biți superiori ai adresei la care se va face un apel indirect de subrutină ca răspuns la o cerere de întrerupere, cei 8 biți inferiori ai adresei fiind furnizați de dispozitivul care a cerut întreruperea, ceea ce creează posibilitatea plasării unei subrutine de deservire a unei întreruperi oriunde în memoria sistemului; registrul de refreșare a memoriei (R), de 7 biți, incrementat după fiecare aducere a unei instrucțiuni din memorie, cuvântul din numărătorul de refreșare fiind trimis pe liniile inferioare ale magistralei de adrese împreună cu un semnal de control al refreșării, în timp ce unitatea centrală decodifică și execută instrucțiunea adusă; în acest mod se realizează refreșarea memoriilor dinamice RAM care pot fi utilizate la fel de simplu ca și cele statice, modul de refreșare fiind transparent pentru utilizator; programatorul poate încărca sau citi registrul R pentru anumite testări, dar în mod uzual acest registru nu este folosit în programe.

Al treilea grup constă din două bistabile pentru controlul întreruperilor, IFF1 și IFF2, și o pereche adițională de bistabile care ajută la identificarea, în orice moment a modului de tratare a întreruperilor.

Registrele unității centrale sînt reprezentate în figura 2.2.



ACUMULATOR A	INDICATORI DE CONDITII F	ACUMULATOR A'	INDICATORI DE CONDITII F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

REGISTRE DE  
UZ GENERAL

VECTOR DE INTRERUPERE I	REFRESARE MEMORIE R
REGISTRU INDEX	IX
REGISTRU INDEX	IY
INDICATOR DE STIVĂ	SP
NUMĂRĂTOR DE PROGRAM	PC

REGISTRE CU  
DESTINATIE  
SPECIALĂ

BISTABILE DE STARE A INTRERUPERII

IFF 1

0 = NU ACCEPȚĂ INTRERUPERI  
1 = ACCEPȚĂ INTRERUPERI

IFF 2

CONTINE  
IFF1 IN  
TIMPUL  
DESERVIRII  
UNEI CERERI NMI

BISTABILE PENTRU MODUL INTRERUPERII

IMF 1 IMF 0

0 0 MOD 0  
0 1 NEUTILIZAT  
1 0 MOD 1  
1 1 MOD 2

Fig. 2.2 Registrele unității centrale.

Unitatea aritmetică/logică (ALU) servește pentru executarea instrucțiunilor aritmetice și logice de 8 biți. Unitatea comunică cu registrele unității centrale și cu magistrala externă de date prin intermediul unei magistrale interne de date.

Funcțiile îndeplinite sînt cele de adunare, scădere, SI logic, SAU logic, SAU-EXCLUSIV, comparare, rotire, deplasare stînga/dreapta, incrementare, decrementare, înscrisere, ștergere sau testare de bit.

Registrul de instrucțiuni preia fiecare instrucțiune adusă din memorie și o decodifică, funcție realizată de secțiunea de control care generează toate semnalele necesare citirii sau scrierii datelor din sau în registrele microprocesorului. Secțiunea de control coordonează și activitatea unității aritmetico-logice și asigură toate semnalele de control extern.

## 2.2. DESCRIEREA FUNCȚIILOR LOGICE ALE CIRCUITULUI Z 80 CPU

Funcțiile logice ale microprocesorului Z 80 sînt prezentate în figura 2.3.

În continuare se dă o descriere a rolului și modului de acțiune a tuturor semnalelor de intrare sau ieșire.

A0—A15 — magistrala de adrese, ieșiri cu trei stări; cele 16 linii formează o adresă de 16 biți pentru schimburi de date cu memoria (max 64 Ko) sau cu dispozitivele de intrare-ieșire (max. 256 DI/E).

BUSACK — (bus acknowledge) ieșire care indică dispozitivului ce solicită acest lucru faptul că magistrala de adrese, de date și semnalele de control MREQ, IORQ, RD și WR au trecut în starea de impedanță ridicată; circuitele externe pot controla acum aceste linii.



**BUSREQ** — (bus request) intrarea cerere de magistrală are o prioritate mai mare decât  $\overline{\text{NMI}}$  și este întotdeauna recunoscută la sfârșitul ciclului de mașină curent; **BUSREQ** forțează magistralele de adrese, de date și semnalele  $\overline{\text{MREQ}}$ ,  $\overline{\text{IORQ}}$ ,  $\overline{\text{RD}}$  și  $\overline{\text{WR}}$  în starea de impedanță ridicată astfel încât alte dispozitive să poată controla aceste linii; semnalul este în mod obișnuit conectat printr-un SI cablat și necesită un rezistor

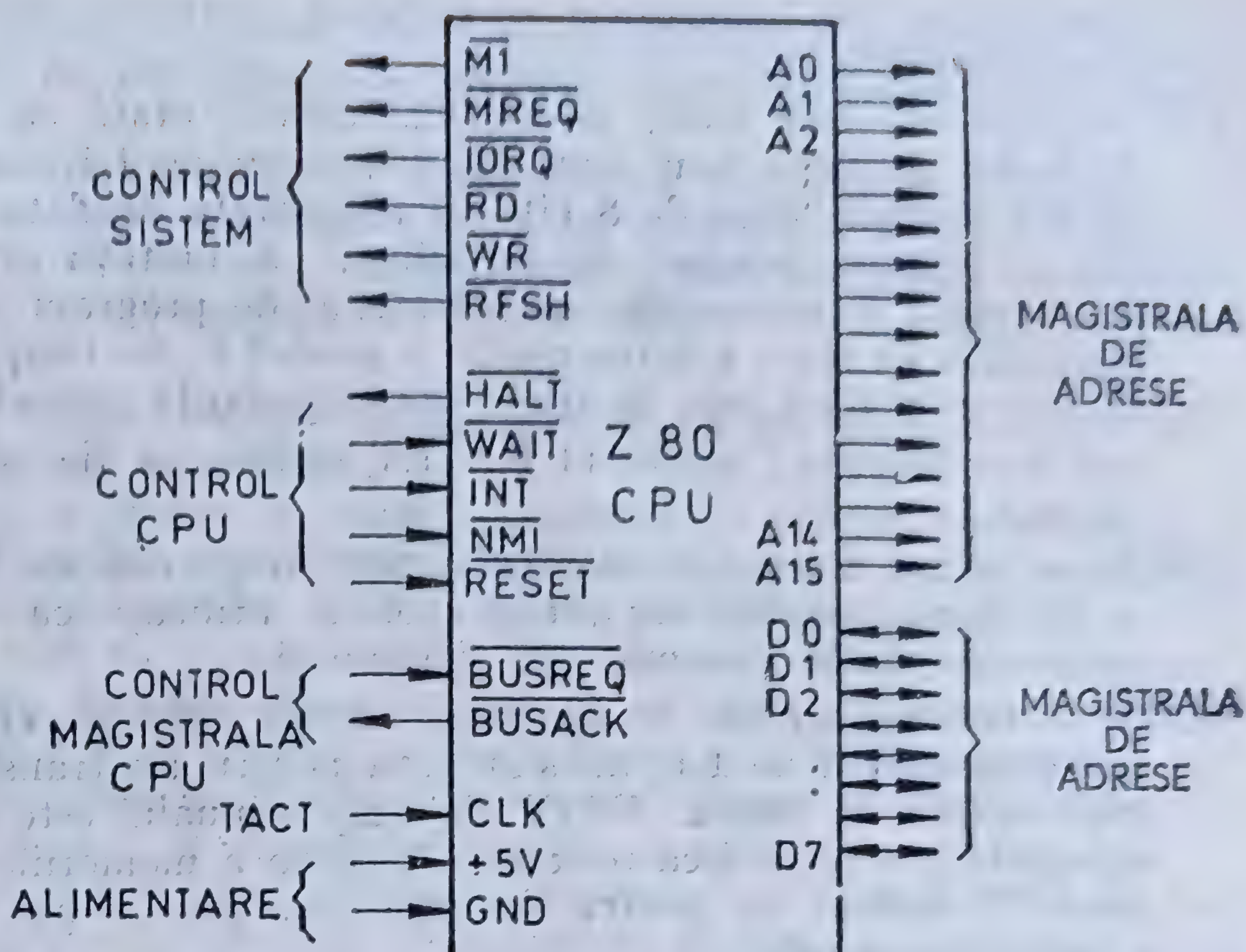


Fig. 2.3 Funcțiile logice ale unității centrale Z80 CPU

exteran la tensiunea pozitivă de alimentare; perioade extinse de zero logic în cazul unor operații de acces direct la memorie, pot împiedica refreșarea corespunzătoare a memoriilor RAM dinamic.

**D0—D7** — intrări/ieșiri, cu trei stări; magistrala de date utilizată pentru schimburi cu memoria și cu dispozitivele de intrare/ieșire.

**$\overline{\text{HALT}}$**  — (halt state), ieșire care indică faptul că unitatea centrală a executat o instrucțiune halt și așteaptă fie o întrerupere pe intrarea  $\overline{\text{NMI}}$ , fie una pe intrarea  $\overline{\text{INT}}$  (bistabilul de acceptare a întreruperii fiind corect poziționat anterior); în timpul stării halt unitatea centrală execută instrucțiunea NOP în mod repetat, pentru a menține refreșarea memoriilor dinamice.

**$\overline{\text{INT}}$**  — intrare, cerere de întrerupere generată de dispozitive de I/E; unitatea centrală deservește întreruperea la sfârșitul instrucțiunii curente, dacă bistabilul intern de acceptare a întreruperilor, IFF, controlat prin program, este poziționat la 1 logic;  $\overline{\text{INT}}$  este în mod obișnuit conectat într-un SI cablat și necesită un rezistor la tensiunea pozitivă de alimentare.

**$\overline{\text{IORQ}}$**  — ieșire cu trei stări, cerere de intrare/ieșire; indică faptul că jumătatea inferioară a magistralei de adrese conține o adresă de I/E pentru o operație de citire sau scriere a unui dispozitiv de I/E; semnalul este de asemenea generat simultan cu  $\overline{\text{MI}}$  în timpul anunțării acceptării unei întreruperi, pentru a arăta că pe magistrala de date poate fi plasat un vector de răspuns.

**$\overline{\text{MI}}$**  — ieșire, primul ciclu de mașină;  $\overline{\text{MI}}$ , împreună cu  $\overline{\text{MREQ}}$  arată că ciclul de mașină curent este cel de aducere a codului unei operații din cadrul executării unei instrucțiuni;  $\overline{\text{MI}}$ , împreună cu  $\overline{\text{IORQ}}$  indică un ciclu de acceptare a unei întreruperi.

**$\overline{\text{MREQ}}$**  — ieșire cu trei stări, cerere de memorie; arată că magistrala de adrese conține o adresă validă pentru o operație de scriere sau citire a memoriei.

**$\overline{\text{NMI}}$**  — intrare, cerere de întrerupere necondiționată; are o prioritate mai mare decât  $\overline{\text{INT}}$ ; este recunoscută la sfârșitul instrucțiunii curente, indiferent de starea



bistabilului de întreruperi și forțează controlul unității centrale la locația de memorie 0066<sub>H</sub>.

$\overline{RD}$  — ieșire cu trei stări, semnal de citire; arată că unitatea centrală va citi o dată din memorie sau dintr-un dispozitiv de I/E, care vor folosi acest semnal pentru a ghida data de 8 biți pe magistrala de date a CPU.

$\overline{RESET}$  — intrare, semnal de inițializare a unității centrale; șterge bistabilul de acceptare a întreruperilor, numărătorul de program, registrele I și R și fixează bistabilele de stare a întreruperii la modul 0; în timpul inițializării, magistralele de adrese și date trec în starea de impedanță ridicată și toate ieșirile de control sînt inactive; semnalul  $\overline{RESET}$  trebuie să fie activ cel puțin trei perioade de tact.

$\overline{RFSH}$  — ieșire, semnal de refreșare, care, împreună cu  $\overline{MREQ}$  arată că biții A/0—A6 ai magistralei de adrese pot fi utilizați ca adresă de refreșare pentru memoriile RAM dinamic ale sistemului.

$\overline{WAIT}$  — intrare, semnal de așteptare; arată unității centrale că memoria adresată sau dispozitivul de I/E nu sînt gata pentru un transfer de date; unitatea centrală se află în starea  $\overline{WAIT}$  cît timp semnalul este activ; perioade extinse de așteptare pot împiedica refreșarea corectă a memoriilor RAM dinamic; semnalul poate fi utilizat și pentru funcționarea pas cu pas (cîte un ciclu de mașină) a microprocesorului.

$\overline{WR}$  — ieșire cu trei stări, semnal de scriere; arată că magistrala de date conține date valide pentru depozitare în locația de memorie sau în DI/E adresat.

$\overline{CLK}$  — intrare, semnalul de tact standard cu o singură fază, al sistemului (2,5 MHz pentru Z 80 CPU).

### 2.3. TRATAREA ÎNTRERUPERILOR

Unitatea centrală acceptă 2 semnale de întrerupere:  $\overline{NMI}$  și  $\overline{INT}$ .

Primul este o cerere de întrerupere necondiționată și are cea mai mare prioritate;  $\overline{INT}$  are prioritate mai mică și trebuie să fie validată prin program pentru a fi acceptată.

Unitatea centrală răspunde într-un singur mod la o cerere  $\overline{NMI}$  și în trei moduri, programabile la o cerere  $\overline{INT}$ : modul 0, compatibil cu microprocesorul 8080; modul 1, pentru servirea întreruperilor de la periferice care nu fac parte din sistemele 8080 sau Z 80; modul 2, pentru lucrul într-o schemă de întreruperi orientată, uzual un lanț de priorități, format din componentele familiei Z 80 și dispozitivele periferice compatibile.

Unitatea centrală deservește întreruperile eșantionînd semnalele  $\overline{NMI}$  și  $\overline{INT}$  pe frontul crescător al ultimului tact dintr-o instrucțiune. Modul de deservire depinde în continuare de tipul întreruperii.

Cererea  $\overline{NMI}$  nu poate fi dezactivată prin program și va fi acceptată întotdeauna. Se utilizează pentru întreruperile cu cea mai mare prioritate. După recunoașterea cererii  $\overline{NMI}$  (dacă  $\overline{BUSREQ}$  nu este activ), execuția programului va trece la locația 0066<sub>H</sub>, la care se află în mod obișnuit începutul rutinei de servire a întreruperii.

Cererea  $\overline{INT}$  are ca urmare o serie de acțiuni în timp care nu depind de modul de întrerupere fixat de utilizator. Dacă întreruperile sînt validate și  $\overline{BUSREQ}$  nu este activ, cererea  $\overline{INT}$  declanșează un ciclu special de aducere a codului operației ( $\overline{MI}$ ), în care  $\overline{IORQ}$  devine activ (și nu  $\overline{MREQ}$ , ca de obicei); în plus acest



ciclu  $\overline{M1}$  este automat extins cu două stări WAIT pentru a da timpul necesar anunțării acceptării cererii de întrerupere și pentru a plasa vectorul de întrerupere pe magistrală.

Modul 0 de tratare este compatibil cu componentele din familia 8080. Dispozitivul care cere întreruperea plasează o instrucțiune pe magistrala de date, executată apoi în 6 perioade de unitatea centrală. Aceasta este în mod obișnuit o instrucțiune Restart, care va iniția un salt necondiționat la una dintre cele 8 locații de restart din pagina 0 de memorie.

Modul 1 de tratare a întreruperilor este asemănător cu cel pentru cererile  $\overline{NMI}$ , cu diferența că adresa de salt este  $0038_H$ .

Modul 2 are rolul de a utiliza la maxim posibilitățile microprocesorului Z 80 și ale componentelor din familia lui. Dispozitivul care a cerut întreruperea alege adresa de început a rutinei de servire a întreruperii, plasând pe magistrala de date un vector de adresă de 8 biți, în timpul ciclului de acceptare a întreruperii. Octetul superior al adresei la care se află adresa rutinei de servire a întreruperii este dat de registrul I. Flexibilitatea în selectarea acestei adrese permite dispozitivului periferic să utilizeze mai multe rutine de deservire care pot fi plasate oriunde în locațiile disponibile de memorie. Bitul 0 al vectorului trebuie să aibă valoarea 0.

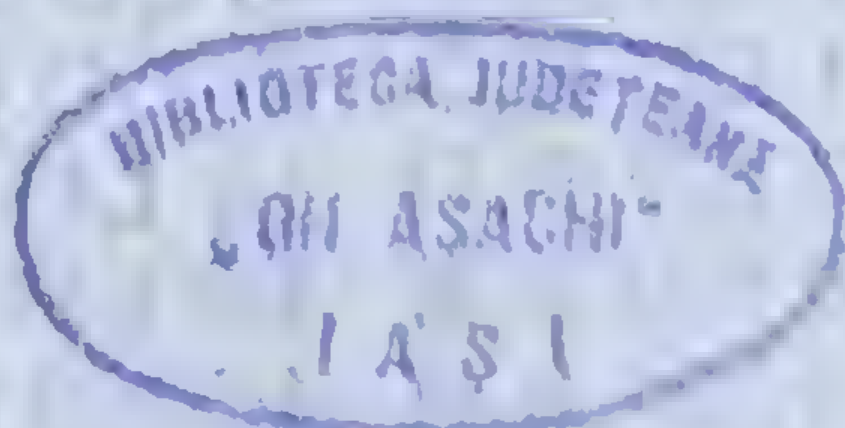
Prioritatea la întreruperi a fiecărui dispozitiv periferic este determinată de poziția fizică a lui într-o configurație de lanț de priorități (daisy-chain). Fiecare dispozitiv din lanț are o intrare de validare a întreruperilor (IEI) și o ieșire de validare a întreruperilor (IEO), conectată la dispozitivul imediat următor, avînd o prioritate mai mică. Primul dispozitiv din lanț are intrarea IEI conectată la 1 logic și are cea mai mare prioritate în timp ce următoarele dispozitive au priorități tot mai scăzute, ceea ce permite unității centrale să selecteze cererea de întrerupere cea mai prioritară în cazul apariției simultane a mai multor cereri. Dispozitivul care a cerut întreruperea își dezactivează linia IEO, pînă la deservirea cererii, după care IEO revine la 1 logic, permițînd dispozitivelor cu un ordin mai mic de prioritate să ceară întreruperi. Unitatea centrală va înregistra cererile apărute în timpul deservirii unei cereri de întrerupere.

Starea întreruperilor este semnalizată unității centrale cu ajutorul a 2 bistabile, IFF1 și IFF2, conform tabelului 2.1.

Starea întreruperilor

Tabelul 2.1.

Acțiunea	IFF1	IFF2	Comentariu
Reset CPU	0	0	Cererile $\overline{INT}$ nevalidate
Execuția instrucțiunii DI	0	0	Cererile $\overline{INT}$ nevalidate
Execuția instrucțiunii EI	1	1	Cererile $\overline{INT}$ validate
Execuția instrucțiunii LD A,I	X	X	IFF2 → bistabilul de paritate
Execuția instrucțiunii LD A,R	X	X	IFF2 → bistabilul de paritate
Acceptare $\overline{NMI}$	0	IFF1	IFF1 → IFF2; cererile $\overline{INT}$ nevalidate
Execuția instrucțiunii RETN	IFF2	X	IFF2 → IFF1, la terminarea rutinei de servire a unei cereri $\overline{NMI}$



543.834



## CAPITOLUL III

### PROGRAMAREA MICROPROCESORULUI Z 80

Se vor prezenta setul de instrucțiuni, modurile de adresare, acțiunea asupra indicatorilor de condiții și exemple de programe utilizând instrucțiunile specifice microprocesorului Z 80.

#### 3.1. SETUL INSTRUȚIUNILOR EXECUTABILE ALE MICROPROCESORULUI Z 80

În descrierea instrucțiunilor se vor utiliza următoarele notații:

n: constantă întreagă de 8 biți

d: deplasament — constantă întreagă de 8 biți reprezentată în aritmetica complementului față de 2, deci cuprinsă în gama  $-128 \div +127$ .

dd,ss: perechea de registre BC, DE, HL sau SP

pp: perechea de registre BC, DE, IX sau SP

rr: perechea de registre BC, DE, IX sau SP

qq: perechea de registre BC, DE, HL sau AF

r: registrul B, C, D, E, H, L sau A

A, B, C, D, E, H, L, A', B', C', D', E', H', L': registrele sau conținutul registrelor de 8 biți cu acest nume

BC, DE, HL, SP, PC, IX, IY: registrele duble sau conținutul registrelor de 16 biți cu acest nume

(BC), (DE), (HL), (SP), (PC), (IX), (IY), (IY+d), (IY+d): conținutul locațiilor de memorie cu adresa conținută în registrele duble respective (adunând și deplasamentul d pentru a obține adresa în ultimele două cazuri)

(A), (C): portul de intrare/ieșire cu adresa conținută în registrul specificat

b: bitul 0, 1, 2, 3, 4, 5, 6 sau 7 al unei locații de memorie sau al unui registru.

#### Grupul instrucțiunilor de transfer de date

Acest grup conține instrucțiuni de transfer a datelor spre sau din registre sau memorie. Conținutul locațiilor din care se transferă datele rămâne neschimbat.

1, LD r, r' Efect:  $r \leftarrow r'$ : conținutul registrului r' este transferat în registrul r.

7FLDA,A	47LDB,A	4FLDC,A	57LDD,A	5FLDE,A	67LDH,A	6FLDL,A
78LDA,B	40LDB,B	48LDC,B	50LDD,B	58LDE,B	60LDH,B	68LDL,B
79LDA,C	41LDB,C	49LDC,C	51LDD,C	59LDE,C	61LDH,C	69LDL,C
7ALDA,D	42LDB,D	4ALDC,D	52LDD,D	5ALDE,D	62LDH,D	6ALDL,D
7BLDA,E	43LDB,E	4BLDC,E	53LDD,E	5BLDE,E	63LDH,E	6BLDL,E
7CLDA,H	44LDB,H	4CLDC,H	54LDD,H	5CLDE,H	64LDH,H	6CLDL,H
7DLD A,L	45LDB,L	4DLD C,L	55LDD,L	5DLD E,L	65LDH,L	6DLD L,L

2, LD r,n Efect:  $r \leftarrow n$ ; întregul de 8 biți n este încărcat în registrul r

3En LD A,n	0En LD C,n	1En LD E,n	2En LD I,n
06n LD B,n	16n LD D,n	26n LD H,n	



3. LD A,I  
ED57      Efect:  $A \leftarrow I$ ; conținutul registrului vectorului de control al întreruperilor este transferat în acumulator.
4. LD I,A  
ED47      Efect:  $I \leftarrow A$ ; conținutul acumulatorului este încărcat în registrul vectorului de control al întreruperilor.
5. LD A,R  
ED5F      Efect:  $A \leftarrow R$ ; conținutul registrului de refreșare a memorie este încărcat în acumulator.
6. LD R,A  
ED4F      Efect:  $R \leftarrow A$ ; conținutul acumulatorului este încărcat în registrul vectorului de refreșare a memoriei.
7. LD (HL),r      Efect:  $(HL) \leftarrow r$ ; conținutul registrului r este încărcat în locația de memorie specificată de conținutul perechii HL.
- 77 LD (HL),A    71 LD (HL),C    73 LD (HL),E    75 LD (HL),L  
70 LD (HL),B    72 LD (HL),D    74 LD (HL),H
8. LD r,(HL)      Efect:  $r \leftarrow (HL)$ ; conținutul locației de memorie cu adresa specificată de conținutul perechii HL este încărcat în registrul r.
- 7E LD A,(HL)    4E LD C,(HL)    5E LD E,(HL)    6E LD L,(HL)  
46 LD B,(HL)    56 LD D,(HL)    66 LD H,(HL)
9. LD (HL),n      Efect:  $(HL) \leftarrow n$ ; întregul n este încărcat la adresa din memorie specificată de conținutul perechii HL.
- 36 n
10. LD A,(BC)      Efect:  $A \leftarrow (BC)$ ; conținutul locației de memorie specificată de conținutul perechii BC este încărcat în acumulator.
- 0A
11. LD (BC),A      Efect:  $(BC) \leftarrow A$ ; transfer în sens invers față de cazul precedent
- 02
12. LD A,(DE)      Efect:  $A \leftarrow (DE)$ ; similar cu instrucțiunea 10 dar locația de memorie specificată de registrul DE
- 1A
13. LD (DE),A      Efect:  $(DE) \leftarrow A$ ; transfer similar celui din instrucțiunea 11, dar locația de memorie este specificată în DE
- 12
14. LD A,(nn)      Efect:  $A \leftarrow (nn)$ ; conținutul locației de memorie specificată de operanzii nn este încărcat în acumulator; primul operand n este octetul de ordin inferior al adresei de 2 octeți.
- 3A n n
15. LD (nn),A      Efect:  $(nn) \leftarrow A$ ; conținutul acumulatorului este încărcat în locația de memorie cu adresa specificată de operanzii nn; primul operand este octetul inferior al adresei.
- 32 n n
16. LD r,(IX + d)      Efect:  $r \leftarrow (IX + d)$ ; conținutul locației de memorie cu adresa IX + d este încărcat în registrul r.
- DD7E d LD A,(IX + d)    DD56 d LD D,(IX + d)    DD6E d LD L,(IX + d)  
DD46 d LD B,(IX + d)    DD5E d LD E,(IX + d)  
DD4E d LD C,(IX + d)    DD66 d LD H,(IX + d)
17. LD (IX + d),r      Efect:  $(IX + d) \leftarrow r$ ; conținutul registrului r este încărcat în locația de memorie cu adresa IX + d.
- DD77 d LD (IX + d),A    DD72 d LD (IX + d),D    DD75 d LD (IX + d),L  
DD70 d LD (IX + d),B    DD73 d LD (IX + d),E  
DD71 d LD (IX + d),C    DD74 d LD (IX + d),H
18. LD r,(IY + d)      Efect:  $r \leftarrow (IY + d)$ ; conținutul locației de memorie cu adresa IY + d este încărcat în registrul r.
- FD7E d LD A,(IY + d)    FD56 d LD D,(IY + d)    FD6E d LD L,(IY + d)  
FD46 d LD B,(IY + d)    FD5E d LD E,(IY + d)  
FD4E d LD C,(IY + d)    FD66 d LD H,(IY + d)
19. LD (IY + d),r      Efect:  $(IY + d) \leftarrow r$ ; conținutul registrului r este încărcat în locația de memorie cu adresa IY + d.
- FD77 d LD (IY + d),A    FD72 d LD (IY + d),D    FD75 d LD (IY + d),L  
FD70 d LD (IY + d),B    FD73 d LD (IY + d),E  
FD71 d LD (IY + d),C    FD74 d LD (IY + d),H
20. LD (IX + d),n      Efect:  $(IX + d) \leftarrow n$ ; întregul n este încărcat în locația de memorie cu adresa IX + d.
- DD36 d n



21. LD (IY+d),n Efect:  $(IY+d) \leftarrow n$ ; întregul n este încărcat în locația de memorie FD36 d n cu adresa (IY+d)
22. LD dd,nn Efect:  $dd \leftarrow nn$ ; întregul de 2 octeți nn este încărcat în perechea de registre dd; primul operand n în codul obiect este octetul inferior.  
 01 n n LD BC,nn 21 n n LD HL,nn  
 11 n n LD DE,nn 31 n n LD SP,nn
23. LD IX,nn Efect:  $IX \leftarrow nn$ ; întregul de doi octeți nn este încărcat în registrul DD21 n n index IX; primul operand n este octetul inferior
24. LD IY,nn Efect:  $IY \leftarrow nn$ ; similar cu instrucțiunea 23, pentru registrul index IY. FD21 n n
25. LD (nn),dd Efect:  $(nn+1) \leftarrow dd_H$ ,  $(nn) \leftarrow dd_L$ ; octetul de ordin inferior al perechii dd este încărcat la adresa de memorie nn, iar octetul superior la adresa nn+1.  
 ED43 n n LD (nn),BC ED63 n n LD (nn),HL  
 ED53 n n LD (nn),DE ED73 n n LD (nn),SP
26. LD (nn),HL Efect:  $(nn+1) \leftarrow H$ ,  $(nn) \leftarrow L$ ; similar cu instrucțiunea 25 dar numai 22 n n pentru perechea HL
27. LD (nn),IX Efect:  $(nn+1) \leftarrow IX_H$ ,  $(nn) \leftarrow IX_L$ ; similar cu instrucțiunea 25 pentru registrul IX. DD22 n n
28. LD (nn),IY Efect:  $(nn+1) \leftarrow IY_H$ ,  $(nn) \leftarrow IY_L$ ; similar cu instrucțiunea 25, pentru registrul IY. FD22 n n
29. LD dd,(nn) Efect:  $dd_H \leftarrow (nn+1)$ ,  $dd_L \leftarrow (nn)$ ; conținutul adresei nn este încărcat în jumătatea inferioară a registrului dublu dd iar conținutul adresei nn+1 este încărcat în jumătatea superioară a lui dd.  
 ED4B n n LD BC,(nn) ED6B n n LD HL,(nn)  
 ED5B n n LD DE,(nn) ED7B n n LD SP,(nn)
30. LD HL,(nn) Efect:  $H \leftarrow (nn+1)$ ,  $L \leftarrow (nn)$ ; similar cu instrucțiunea 29, dar numai 2A n n pentru registrul dublu HL
31. LD IX,(nn) Efect:  $IX_H \leftarrow (nn+1)$ ,  $IX_L \leftarrow (nn)$ ; similar cu instrucțiunea 29, dar DD2A n n pentru registrul IX
32. LD IY,(nn) Efect:  $IY_H \leftarrow (nn+1)$ ,  $IY_L \leftarrow (nn)$ ; similar cu instrucțiunea 29, dar FD2A n n pentru registrul IY
33. LD SP,HL Efect:  $SP \leftarrow HL$ ; conținutul perechii de registre HL este încărcat în registrul indicator de stivă SP F9
34. LD SP,IX Efect:  $SP \leftarrow IX$ ; conținutul registrului index IX este încărcat în registrul indicator de stivă SP DDF9
35. LD SP,IY Efect:  $SP \leftarrow IY$ ; conținutul registrului index IY este încărcat în registrul indicator de stivă SP FDF9
36. LDD EDA8 Efect:  $(DE) \leftarrow (HL)$ ,  $DE \leftarrow DE-1$ ,  $HL \leftarrow HL-1$ ,  $BC \leftarrow BC-1$ ; se transferă un octet de date din locația de memorie adresată de conținutul perechii de registre HL, în locația de memorie adresată de conținutul perechii de registre DE; perechile de registre HL, DE și BC sînt decrementate; BC poate fi utilizat ca numărator de octeți.
37. LDDR EDB8 Efect:  $(DE) \leftarrow (HL)$ ,  $DE \leftarrow DE-1$ ,  $HL \leftarrow HL-1$ ,  $BC \leftarrow BC-1$ , pînă cînd  $(BC)=0$ ; se încarcă un octet din locația de memorie adresată de conținutul perechii de registre HL, în locația de memorie adresată de conținutul perechii de registre DE; în continuare perechile de registre HL, DE și BC (numărator de octeți) sînt decrementate; dacă prin decrementare conținutul lui BC devine 0, instrucțiunea este terminată; dacă nu devine 0, numărator de program este decrementat cu 2 și instrucțiunea se repetă; dacă BC conține 0 înaintea executării instrucțiunii, atunci se vor transfera 64 de kiloocteți de date; întreruperile se recunosc și două cicluri de refreșare se execută după fiecare transfer de date.



38. LDI EDA0 Efect:  $(DE) \leftarrow (HL)$ ,  $DE \leftarrow DE + 1$ ,  $HL \leftarrow HL + 1$ ,  $BC \leftarrow BC - 1$ ; similar cu instrucțiunea 36, dar perechile de registre DE și HL sînt incrementate după transfer.
39. LDIR EDB0 Efect:  $(DE) \leftarrow (HL)$ ,  $DE \leftarrow DE + 1$ ,  $HL \leftarrow HL + 1$ ,  $BC \leftarrow BC - 1$ , pînă cînd  $(BC) = 0$ ; similar cu instrucțiunea 37 dar perechile de registre DE și HL sînt incrementate după fiecare transfer.
40. EX DE,HL EB Efect:  $DE \leftrightarrow HL$ ; conținutul de 2 octeți al perechii de registre DE este schimbat cu conținutul perechii HL;
41. EX AF,AF' 08 Efect:  $AF \leftrightarrow AF'$ ; conținutul de 2 octeți al perechii de registre AF este schimbat cu conținutul perechii AF' (formată din A',F')
42. EXX D9 Efect:  $BC \leftrightarrow BC'$ ,  $DE \leftrightarrow DE'$ ,  $HL \leftrightarrow HL'$ ; conținutul perechilor de registre BC, DE, HL este schimbat, respectiv, cu conținutul perechilor BC', DE', HL' (formate din B', C', D', E', H', L')

### Grupul instrucțiunilor aritmetice

Acest grup realizează operații aritmetice asupra datelor din registre și memorie: adunări și scăderi de 1 sau 2 octeți, incrementări, decrementări, operații BCD.

43. ADD A,r Efect:  $A \leftarrow A + r$ ; conținutul registrului r este adunat la conținutul acumulatorului și rezultatul este plasat în acumulator.
- 87 ADD A,A    81 ADD A,C    83 ADD A,E    85 ADD A,L  
80 ADD A,B    82 ADD A,D    84 ADD A,H
44. ADD A,n C6 n Efect:  $A \leftarrow A + n$ ; întregul n este adunat la conținutul acumulatorului și rezultatul este plasat în acumulator.
45. ADD A,(HL) 86 Efect:  $A \leftarrow A + (HL)$ ; octetul din locația de memorie adresată de conținutul registrului dublu HL este adunat la conținutul acumulatorului și rezultatul este plasat în acumulator.
46. ADD A,(IX+d) DD86 d Efect:  $A \leftarrow A + (IX + d)$ ; conținutul registrului index IX este adunat la deplasamentul d pentru a indica o locație de memorie; conținutul acestei locații este adunat la conținutul acumulatorului și rezultatul este plasat în acumulator.
47. ADD A,(IY+d) FD86 d Efect:  $A \leftarrow A + (IY + d)$ ; similar cu instrucțiunea 46, dar pentru registrul index IY.
48. ADC A,r Efect:  $A \leftarrow A + r + CY$ ; conținutul registrului r și conținutul indicatorului de transport (bitul C din registrul F) sînt adunate la conținutul acumulatorului, iar rezultatul este plasat în acumulator.
- 88 ADC A,B    8A ADC A,D    8C ADC A,H    8F ADC A,A  
89 ADC A,C    8B ADC A,E    8D ADC A,L
49. ADC A,n CE n Efect:  $A \leftarrow A + n + CY$ ; întregul n și conținutul indicatorului de transport sînt adunate la conținutul acumulatorului, iar rezultatul este plasat în acumulator.
50. ADC A,(HL) 8E Efect:  $A \leftarrow A + (HL) + CY$ ; octetul din locația de memorie adresată de conținutul registrului dublu HL și conținutul indicatorului de transport sînt adunate la conținutul acumulatorului iar rezultatul este plasat în acumulator.
51. ADC A,(IX+d) DD8E d Efect:  $A \leftarrow A + (IX + d) + CY$ ; conținutul registrului index IX este adunat la deplasamentul d pentru a indica o locație de memorie; conținutul ei și conținutul indicatorului de transport sînt adunate la conținutul acumulatorului, iar rezultatul este plasat în acumulator.
52. ADCA, (IY+d) FD8E d Efect:  $A \leftarrow A + (IY + d) + CY$ ; similar cu instrucțiunea 51, dar pentru registrul index IY.



53. ADD HL,ss Efect:  $HL \leftarrow HL + ss$ ; conținutul perechii de registre ss este adunat la conținutul perechii de registre HL și rezultatul este plasat în registrul dublu HL.

09 ADD HL,BC      29 ADD HL,HL

19 ADD HL,DE      39 ADD HL,SP

54. ADC HL,ss Efect:  $HL \leftarrow HL + ss + CY$ ; conținutul perechii de registre ss și conținutul indicatorului de transport (bitul C din registrul F) sînt adunate la conținutul perechii de registre HL și rezultatul este plasat în registrul dublu HL.

ED4A      ADC HL,BC      ED6A      ADC HL,HL

ED5A      ADC HL,DE      ED7A      ADC HL,SP

55. ADD IX,pp Efect:  $IX \leftarrow IX + pp$ ; conținutul perechii de registre pp este adunat la conținutul registrului index IX iar rezultatul este plasat în registrul index IX.

DD09      ADD IX,BC      DD29      ADD IX,IX

DD19      ADD IX,DE      DD39      ADD IX,SP

56. ADD IY,rr Efect:  $IY \leftarrow IY + rr$ ; conținutul perechii de registre rr este adunat la conținutul registrului index IY, iar rezultatul este plasat în registrul index IY.

FD09      ADD IY,BC      FD29      ADD IY,IY

FD19      ADD IY,DE      FD39      ADD IY,SP

57. SUB r Efect:  $A \leftarrow A - r$ ; conținutul registrului r este scăzut din conținutul acumulatorului iar rezultatul este plasat în acumulator.

90 SUB B      92 SUB D      94 SUB H      97 SUB A

91 SUB C      93 SUB E      95 SUB L

58. SUB n Efect:  $A \leftarrow A - n$ ; întregul n este scăzut din conținutul acumulatorului iar rezultatul este plasat în acumulator.

D6 n

59. SUB (HL) Efect:  $A \leftarrow A - (HL)$ ; octetul din locația de memorie adresată de conținutul registrului dublu HL este scăzut din conținutul acumulatorului iar rezultatul este plasat în acumulator.

96

60. SUB (IX+d) Efect:  $A \leftarrow A - (IX+d)$ ; similar cu instrucțiunea 59, dar locația de memorie are adresa IX+d.

DD96 d

61. SUB (IY+d) Efect:  $A \leftarrow A - (IY+d)$ ; similar cu instrucțiunea 59 dar locația de memorie are adresa IY+d.

FD96 d

62. SBC A,r Efect:  $A \leftarrow A - r - CY$ ; conținutul registrului r și conținutul indicatorului de transport sînt scăzute din conținutul acumulatorului, iar rezultatul este plasat în acumulator.

98 SBC A,B      9A SBC A,D      9C SBC A,H      9F SBC A,A

99 SBC A,C      9B SBC A,E      9D SBC A,L

63. SBC A,n Efect:  $A \leftarrow A - n - CY$ ; întregul n și conținutul indicatorului de transport sînt scăzute din conținutul acumulatorului, iar rezultatul este plasat în acumulator.

DE n

64. SBC A,(HL) Efect:  $A \leftarrow A - (HL) - CY$ ; octetul din locația de memorie adresată de conținutul registrului dublu HL și conținutul indicatorului de transport sînt scăzute din conținutul acumulatorului iar rezultatul este plasat în acumulator.

9E

65. SBC A,(IX+d) Efect:  $A \leftarrow A - (IX+d) - CY$ ; similar cu instrucțiunea 64 dar locația de memorie are adresa IX+d.

DD9E d

66. SBC A,(IY+d) Efect:  $A \leftarrow A - (IY+d) - CY$ ; similar cu instrucțiunea 64 dar locația de memorie are adresa IY+d.

FD9E d



67. SBC HL,ss Efect:  $HL \leftarrow HL - ss - CY$ ; conținutul perechii de registre ss și conținutul indicatorului de transport sînt scăzute din conținutul perechii de registre HL, iar rezultatul este plasat în HL.
- ED42 SBC HL,BC ED62 SBC HL,HL  
ED52 SBC HL,DE ED72 SBC HL,SP
68. INC r Efect:  $r \leftarrow r + 1$ ; conținutul registrului r este incrementat cu o unitate.
- 3C INC A 0C INC C 1C INC E 2C INC L  
04 INC B 14 INC D 24 INC H
69. INC (HL) Efect:  $(HL) \leftarrow (HL) + 1$ ; octetul conținut în locația de memorie cu adresa HL este incrementat.
70. INC (IX+d) Efect:  $(IX+d) \leftarrow (IX+d) + 1$ ; similar cu instrucțiunea 69 dar locația de memorie are adresa IX+d.
71. INC (IY+d) Efect:  $(IY+d) \leftarrow (IY+d) + 1$ ; similar cu instrucțiunea 69 dar locația de memorie are adresa IY+d.
72. INC ss Efect:  $ss \leftarrow ss + 1$ ; conținutul perechii de registre ss este incrementat cu o unitate.
- 03 INC BC 13 INC DE 23 INC HL 33 INC SP
73. INC IX Efect:  $IX \leftarrow IX + 1$ ; conținutul registrului index IX este incrementat cu o unitate.
- DD23
74. INC IY Efect:  $IY \leftarrow IY + 1$ ; conținutul registrului index IY este incrementat cu o unitate.
- FD23
75. DEC r Efect:  $r \leftarrow r - 1$ ; conținutul registrului r este decrementat cu o unitate.
- 05 DEC B 15 DEC D 25 DEC H 3D DEC A  
0D DEC C 1D DEC E 2D DEC L
76. DEC (HL) Efect:  $(HL) \leftarrow (HL) - 1$ ; octetul conținut în locația de memorie cu adresa specificată în registrul dublu HL este decrementat cu o unitate.
77. DEC (IX+d) Efect:  $(IX+d) \leftarrow (IX+d) - 1$ ; similar cu instrucțiunea 76 dar locația de memorie are adresa IX+d.
- DD35 d
78. DEC (IY+d) Efect:  $(IY+d) \leftarrow (IY+d) - 1$ ; similar cu instrucțiunea 76 dar locația de memorie are adresa IY+d.
- FD35 d
79. DEC ss Efect:  $ss \leftarrow ss - 1$ ; conținutul perechii de registre ss este decrementat cu o unitate.
- 0B DEC BC 1B DEC DE 2B DEC HL 3B DEC SP
80. DEC IX Efect:  $IX \leftarrow IX - 1$ ; conținutul registrului index IX este decrementat cu o unitate.
- DD2B
81. DEC IY Efect:  $IY \leftarrow IY - 1$ ; conținutul registrului index IY este decrementat cu o unitate.
- FD2B
82. NEG Efect: conținutul acumulatorului primește semn opus, în aritmetica complementului față de 2, ceea ce echivalează cu scăderea conținutului acumulatorului din zero.
- ED44
83. DAA Efect: ajustarea zecimală a acumulatorului; instrucțiunea ajustează conținutul acumulatorului pentru operații de adunare și scădere în cod BCD; tabelul de mai jos indică operația efectuată pentru adunare (ADD, ADC, INC) sau scădere (SUB, SBC, DEC, NEG).
- 27



Operație	C înainte de DAA	Valoare hexa în cifra sup. (Bit 7-4)	II înainte de DAA	Valoare hexa în cifra inf. (Bit 3-0)	Număr adunat la octet	C după DAA
ADD ADC INC	0	0-9	0	0-9	00	0
	0	0-9	0	A-F	06	0
	0	0-9	1	0-3	06	0
	0	A-F	0	0-9	60	1
	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB	0	0-9	0	0-9	00	0
SBC	0	0-9	1	6-F	FA	0
DEC	1	7-F	0	0-9	A0	1
NEG	1	6-F	1	6-F	9A	1

### Grupul instrucțiunilor logice

Acest grup realizează operații logice cu datele din registre, memorie sau din indicatorii de condiții.

84. AND r Efect:  $A \leftarrow A \wedge r$ ; se efectuează operația SI logic, bit cu bit, între conținutul registrului r și conținutul acumulatorului, iar rezultatul se plasează în acumulator

A0 AND B A2 AND D A4 AND H A7 AND A  
A1 AND C A3 AND E A5 AND L

85. AND n E6 n Efect:  $A \leftarrow A \wedge n$ ; se efectuează operația SI logic, bit cu bit, între numărul n și conținutul acumulatorului.

86. AND (HL) A6 Efect:  $A \leftarrow A \wedge (HL)$ ; se efectuează operația SI logic bit cu bit, între conținutul locației de memorie adresată de conținutul registrului dublu HL și conținutul acumulatorului, iar rezultatul este plasat în acumulator.

87. AND (IX+d) DDA6 d Efect:  $A \leftarrow A \wedge (IX+d)$ ; similar cu instrucțiunea 86, dar locația de memorie are adresa IX+d.

88. AND (IY+d) FDA6 d Efect:  $A \leftarrow A \wedge (IY+d)$ ; similar cu instrucțiunea 86, dar locația de memorie are adresa IY+d.

89. OR r Efect:  $A \leftarrow A \vee r$ ; se efectuează operația SAU logic, bit cu bit, între conținutul registrului r și conținutul acumulatorului, iar rezultatul este plasat în acumulator.

B0 OR B B2 OR D B4 OR H B7 OR A  
B1 OR C B3 OR E B5 OR L

90. OR n F6 n Efect:  $A \leftarrow A \vee n$ ; se efectuează operația SAU logic, bit cu bit, între numărul n și conținutul acumulatorului.

91. OR (HL) B6 Efect:  $A \leftarrow A \vee (HL)$ ; se efectuează operația SAU logic, bit cu bit, între conținutul locației de memorie adresată de conținutul registrului dublu HL și conținutul acumulatorului, iar rezultatul este plasat în acumulator.

92. OR (IX+d) DDB6 d Efect:  $A \leftarrow A \vee (IX+d)$ ; similar cu instrucțiunea 91, dar locația de memorie are adresa IX+d.

93. OR (IY+d) FDB6 d Efect:  $A \leftarrow A \vee (IY+d)$ ; similar cu instrucțiunea 91, dar locația de memorie are adresa IY+d.



94. XOR r      Efect:  $A \leftarrow A \oplus r$ ; se efectuează operația SAU EXCLUSIV, bit cu bit, între conținutul registrului r și conținutul acumulatorului, iar rezultatul este plasat în acumulator.
- A8 XOR B      AA XOR D      AC XOR H      AF XOR A  
A9 XOR C      AB XOR E      AD XOR L
95. XOR n      Efect:  $A \leftarrow A \oplus n$ ; se efectuează operația SAU EXCLUSIV, bit cu bit, între numărul n și conținutul acumulatorului.
- EE n
96. XOR (HL)      Efect:  $A \leftarrow A \oplus (HL)$ ; se efectuează operația SAU EXCLUSIV, bit cu bit, între conținutul locației de memorie adresată de conținutul registrului dublu HL și conținutul acumulatorului, iar rezultatul este plasat în acumulator.
- AE
97. XOR(IX+d)      Efect:  $A \leftarrow A \oplus (IX+d)$ ; similar cu instrucțiunea 96, dar locația DDAE d de memorie are adresa IX+d.
- DDAE d
98. XOR (IY+d)      Efect:  $A \leftarrow A \oplus (IY+d)$ ; similar cu instrucțiunea 96, dar locația FDAE d de memorie are adresa IY+d.
- FDAE d
99. CPL      Efect:  $A \leftarrow \bar{A}$ ; conținutul acumulatorului este complementat (complement față de 1).
- 2F
100. CP r      Efect:  $A \div r$ ; conținutul registrului r este comparat cu conținutul acumulatorului prin efectuarea scăderii  $A - r$ .
- B8      CP B      BA CP D      BC CP H      BF CP A  
B9      CP C      BB CP E      BD CP L
101. CP n      Efect:  $A \div n$ ; numărul n este comparat cu conținutul acumulatorului, prin efectuarea scăderii  $A - n$ ; indicatorii de condiții sînt poziționați în funcție de rezultatul comparării.
- FE n
102. CP (HL)      Efect:  $A \div (HL)$ ; conținutul locației de memorie adresată de conținutul registrului dublu HL este comparat cu conținutul acumulatorului prin efectuarea scăderii  $A - (HL)$ ; indicatorii de condiții sînt poziționați în funcție de rezultatul comparării.
- BE
103. CP (IX+d)      Efect:  $A \div (IX+d)$ ; similar cu instrucțiunea 102, dar locația de DDBE d de memorie are adresa IX+d.
- DDBE d
104. CP (IY+d)      Efect:  $A \div (IY+d)$ ; similar cu instrucțiunea 102, dar locația de FDBE d de memorie are adresa IY+d.
- FDBE d
105. CPD      Efect:  $A \div (HL)$ ,  $HL \leftarrow HL - 1$ ,  $BC \leftarrow BC - 1$ ; conținutul locației de EDA9 de memorie adresată de conținutul registrului dublu HL este comparat cu conținutul acumulatorului prin efectuarea scăderii  $A - (HL)$ ; indicatorii de condiții sînt poziționați în funcție de rezultatul comparării; registrul dublu HL și registrul dublu BC (numărătorul de octeți) sînt decrementate.
106. CPDR      Efect:  $A \div (HL)$ ,  $HL \leftarrow HL - 1$ ,  $BC \leftarrow BC - 1$ , pînă cînd  $BC = 0$  sau EDB9  $A = (HL)$ ; conținutul locației de memorie adresată de conținutul registrului dublu HL este comparat cu conținutul acumulatorului prin efectuarea scăderii  $A - (HL)$ ; indicatorii de condiții sînt poziționați în funcție de rezultatul comparării; registrul dublu HL și registrul dublu BC (numărătorul de octeți) sînt decrementate; dacă prin decrementare se obține  $BC = 0$  sau dacă  $A = (HL)$ , instrucțiunea este terminată; dacă  $BC \neq 0$  și  $A \neq (HL)$ , numărătorul de program este decrementat cu 2 și instrucțiunea este repetată; de notat că pentru  $BC = 0$  înaintea executării instrucțiunii, aceasta va efectua compararea pentru 64 de kiloocteți de date dacă nu se îndeplinește condiția  $A = (HL)$ ; întreruperile sînt recunoscute și două cicluri de refreșare a memoriei se execută după fiecare comparare de date.
107. CP      Efect:  $A \div (HL)$ ;  $HL \leftarrow HL + 1$ ,  $BC \leftarrow BC - 1$ ; similar cu instrucțiunea EDA1 105, dar registrul dublu HL este incrementat după transfer.
- EDA1



108. CPiR EDB1 Efect:  $A \div (HL)$ ,  $HL \leftarrow HL + 1$ ,  $BC \leftarrow BC - 1$ , pînă cînd  $BC = 0$  sau  $A = (HL)$ ; similar cu instrucțiunea 106, dar perechea de registre HL este incrementată după fiecare transfer.
109. RL r Efect: conținutul registrului r este rotit spre stînga cu un bit; conținutul bitului 7 este copiat în bitul indicator de transport, iar conțin-

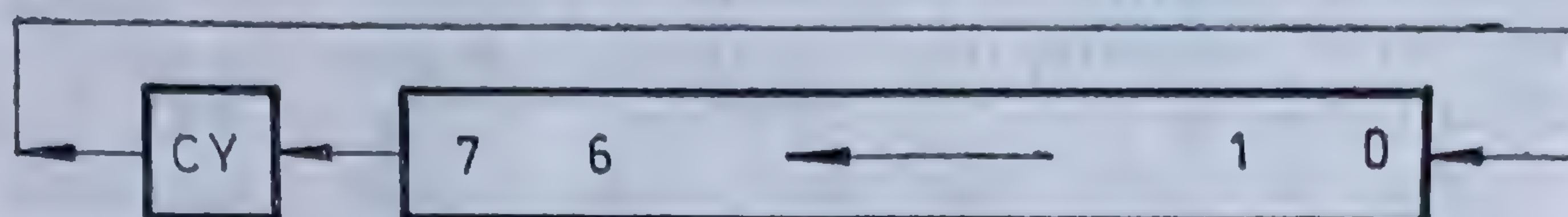


Fig. 3.1 Efectul instrucțiunilor RL r, RL (HL), RL (IX+d), RL (IY+d), RLA.

nutul anterior al bitului indicator de transport este copiat în bitul 0 (cel mai puțin semnificativ bit) v. fig. 3.1.

- |      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
| CB10 | RL B | CB12 | RL D | CB14 | RL H | CB17 | RL A |
| CB11 | RL C | CB13 | RL E | CB15 | RL L |      |      |
110. RL (HL) CB16 Efect: conținutul locației de memorie adresată de registrul dublu HL este rotit spre stînga cu un bit; conținutul bitului 7 este copiat în bitul indicator de transport iar conținutul anterior al acestuia este copiat în bitul 0 (v. figura 3.1).
111. RL (IX+d) Efect: similar cu instrucțiunea 110 dar locația de memorie are DDCB d 16 adresa IX+d (v. fig. 3.1).
112. RL (IY+d) Efect: similar cu instrucțiunea 110, dar locația de memorie are FDCB d16 adresa IY+d (v. fig. 3.1).
113. RLA 17 Efect: similar cu instrucțiunea 109, dar pentru conținutul acumulatorului (v. fig. 3.1).
114. RLC r Efect: conținutul registrului r este rotit spre stînga un bit; conținutul bitului 7 este copiat în indicatorul de transport și în bitul 0 (v. fig. 3.2).
- |      |       |      |       |      |       |      |       |
|------|-------|------|-------|------|-------|------|-------|
| CB00 | RLC B | CB02 | RLC D | CB04 | RLC H | CB07 | RLC A |
| CB01 | RLC C | CB03 | RLC E | CB05 | RLC L |      |       |

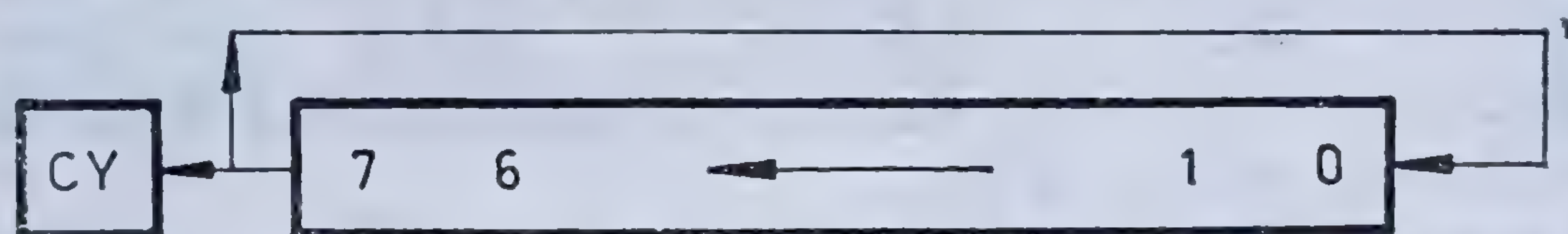


Fig. 3.2 Efectul instrucțiunilor RLC r, RLC (HL), RLC (IX+d), RLC (IY+d), RLCA.

115. RLC (HL) CB06 Efect: conținutul registrului (HL) este rotit spre stînga cu un bit; conținutul bitului 7 este copiat în indicatorul de transport și în bitul 0 (v. fig. 3.2).
116. RLC (IX+d) Efect: similar cu instrucțiunea 115, dar locația de memorie are DDCB d06 adresa IX+d (v. fig. 3.2).
117. RLC (IY+d) Efect: similar cu instrucțiunea 115, dar locația de memorie are FDCB d06 adresa IY+d (v. fig. 3.2).
118. RLCA 07 Efect: similar cu instrucțiunea 114, dar numai pentru conținutul acumulatorului (v. fig. 3.2).
119. RR r Efect: conținutul registrului r este rotit spre dreapta; conținutul bitului 0 este copiat în bitul indicator de transport, iar conținutul anterior al acestuia este copiat în bitul 7. (v. fig. 3.3).
- |      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|
| CB18 | RR B | CB1A | RR D | CB1C | RR H | CB1F | RR A |
| CB19 | RR C | CB1B | RR E | CB1D | RR L |      |      |



Fig. 3.3 Efectul instrucțiunilor RR r, RR (HL), RR (IX+d), RR (IY+d), RRA.



120. RR (HL) CB1E Efect: conținutul locației de memorie adresată de registrul dublu HL este rotit spre dreapta cu un bit; conținutul bitului 0 este copiat în bitul indicator de transport, iar conținutul anterior al acestuia este copiat în bitul 7 (v. fig. 3.3).
121. RR (IX+d) DDCB d 1E Efect: similar cu instrucțiunea 120, dar locația de memorie are adresa IX+d (v. fig. 3.3).
122. RR (IY+d) FDCB d 1E Efect: similar cu instrucțiunea 120, dar locația de memorie are adresa IY+d (vezi fig. 3.3).
123. RRA 1F Efect: similar cu instrucțiunea 119, dar numai pentru conținutul acumulatorului (vezi fig. 3.3).
124. RRC r Efect: conținutul registrului r este rotit spre dreapta cu un bit; conținutul bitului 0 este copiat în bitul 7 și în bitul indicator de transport (v. fig. 3.4).

CB08 RRC B CB0A RRC D CB0C RRC H CB0F RRC A  
CB09 RRC C CB0B RRC E CB0D RRC L



Fig. 3.4 Efectul instrucțiunilor RRC r, RRC (IX+d), RRC (IY+d), RRCA.

125. RRC (HL) CB0E Efect: conținutul locației de memorie adresată de conținutul registrului HL este rotit spre dreapta cu un bit; conținutul bitului 0 este copiat în bitul 7 și în bitul indicator de transport (vezi fig. 3.4).
126. RRC (IX+d) DDCB d0E Efect: similar cu instrucțiunea 125, dar locația de memorie are adresa IX+d (v. fig. 3.4).
127. RRC(IY+d) FDCB d0E Efect: similar cu instrucțiunea 125, dar locația de memorie are adresa IY+d (v. fig. 3.4).
128. RRCA 0F Efect: similar cu instrucțiunea 124, dar numai pentru conținutul acumulatorului (v. fig. 3.4).
129. RLD ED6F Efect: conținutul celor 4 biți mai puțin semnificativi ai locației de memorie adresată de HL este copiat în cei 4 biți mai semnificativi ai aceleiași locații; conținutul anterior al acestora este copiat în cei 4 biți mai semnificativi ai acumulatorului A, iar conținutul anterior al acestora este copiat în cei 4 biți mai puțin semnificativi ai locației (HL) (v. fig. 3.5).

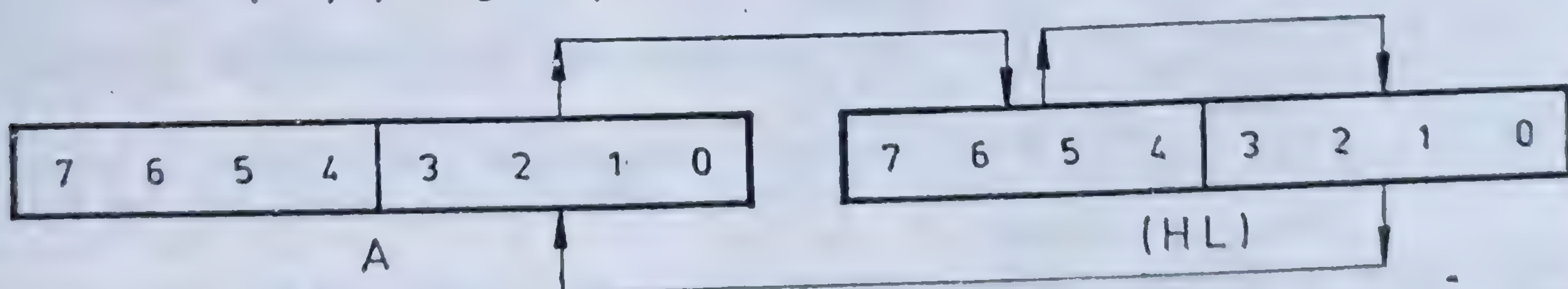


Fig. 3.5 Efectul instrucțiunii RLD.

130. RRD ED67 Efect: asemănător cu al instrucțiunii 129, dar deplasarea are loc spre dreapta, conform figurii 3.6.

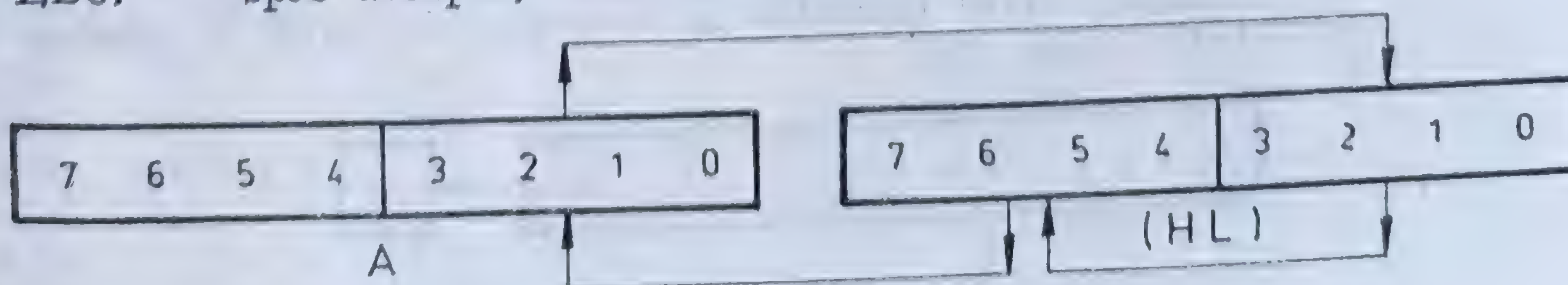


Fig. 3.6 Efectul instrucțiunii RRD.



131. SLA r Efect: se efectuează deplasarea aritmetică spre stînga, cu un bit a conținutului registrului r: bitul 0 este șters, iar bitul 7 este copiat în bitul indicator de transport (v. fig. 3.7).

CB20 SLA B CB22 SLA D CB24 SLA H CB27 SLA A  
CB21 SLA C CB23 SLA E CB25 SLA L



Fig. 3.7 Efectul instrucțiunilor SLA r, SLA (HL), SLA (IX+d), SLA (IY+d).

132. SLA (HL) Efect: se efectuează deplasarea aritmetică spre stînga cu un bit, a conținutului locației de memorie cu adresa în registrul dublu HL; bitul 0 este șters, iar bitul 7 este copiat în bitul indicator de transport (v. fig. 3.7).

133. SLA (IX+d) Efect: similar cu instrucțiunea 132, dar locația de memorie are DDCB d26 adresa IX+d (v. fig. 3.7).

134. SLA (IY+d) Efect: similar cu instrucțiunea 132, dar locația de memorie are FDCB d26 adresa IY+d (v. fig. 3.7).

135. SRL r Efect: se efectuează deplasarea aritmetică spre dreapta, cu un bit, a conținutului registrului r, bitul 7 este șters, iar bitul 0 este copiat în bitul indicator de transport (v. fig. 3.8).

CB38 SRL B CB3A SRL D CB3C SRL H CB3F SRL A  
CB39 SRL C CB3B SRL E CB3D SRL L



Fig. 3.8 Efectul instrucțiunilor SRL r, SRL (HL), SRL (IX+d), SRL (IY+d).

136. SRL (HL) Efect: se efectuează deplasarea aritmetică spre dreapta, cu un bit, a conținutului locației de memorie cu adresa în registrul dublu HL; bitul 7 este șters, iar bitul 0 este copiat în bitul indicator de transport (v. fig. 3.8).

137. SRL (IX+d) Efect: similar cu instrucțiunea 136, dar locația de memorie are DDCB d3E adresa IX+d (v. fig. 3.8).

138. SRL (IY+d) Efect: similar cu instrucțiunea 136, dar locația de memorie are FDCB d3E adresa IY+d (v. fig. 3.8)

139. SRA r Efect: se efectuează deplasarea aritmetică spre dreapta a registrului r; bitul 0 este copiat în bitul indicator de transport, iar bitul 7 rămîne neschimbat (v. fig. 3.9).

CB28 SRA B CB2A SRA D CB2C SRA H CB2F SRA A  
CB29 SRA C CB2B SRA E CB2D SRA L

140. SRA (HL) Efect: se efectuează deplasarea aritmetică spre dreapta a conținutului locației de memorie cu adresa în registrul dublu HL; bitul 0 este copiat în bitul indicator de transport iar bitul 7 rămîne neschimbat (v. fig. 3.9).

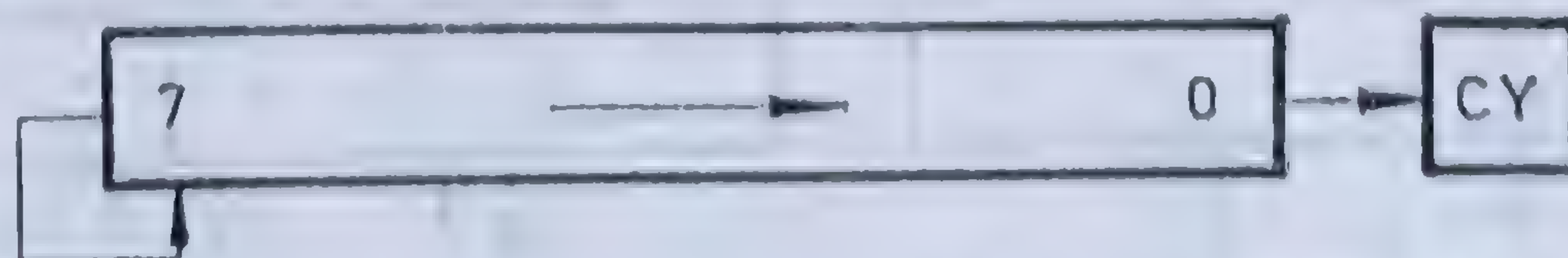


Fig. 3.9 Efectul instrucțiunilor SRA r, SRA (HL), SRA (IX+d), SRA (IY+d).



141. SRA (IX+d) Efect: similar cu instrucțiunea 140, dar adresa locației de memorie DDCB d2E este IX+d (v. fig. 3.9).

142. SRA (IY+D) Efect: similar cu instrucțiunea 140, dar locația de memorie are FDCB d2E adresa IY+d (v. fig. 3.9).

143. SCF Efect:  $C \leftarrow 1$ ; se înscrie bitul C din registrul F (cu valoarea 1 logic).

144. CCF Efect:  $C \leftarrow \bar{C}$ ; bitul C din registrul F este complementat (complement față de 1).

145. BIT b,r Efect:  $Z \leftarrow \bar{r}_b$ ; bitul Z registrul F este înscris cu complementul față de 1 al bitului b din registrul r.

CB40	BIT 0,B	CB42	BIT 0,D	CB44	BIT 0,H	CB47	BIT 0,A
CB48	BIT 1,B	CB4A	BIT 1,D	CB4C	BIT 1,H	CB4F	BIT 1,A
CB50	BIT 2,B	CB52	BIT 2,D	CB54	BIT 2,H	CB57	BIT 2,A
CB58	BIT 3,B	CB5A	BIT 3,D	CB5C	BIT 3,H	CB5F	BIT 3,A
CB60	BIT 4,B	CB62	BIT 4,D	CB64	BIT 4,H	CB67	BIT 4,A
CB68	BIT 5,B	CB6A	BIT 5,D	CB6C	BIT 5,H	CB6F	BIT 5,A
CB70	BIT 6,B	CB72	BIT 6,D	CB74	BIT 6,H	CB77	BIT 6,A
CB78	BIT 7,B	CB7A	BIT 7,D	CB7C	BIT 7,H	CB7F	BIT 7,A
CB41	BIT 0,C	CB43	BIT 0,E	CB45	BIT 0,L		
CB49	BIT 1,C	CB4B	BIT 1,E	CB4D	BIT 1,L		
CB51	BIT 2,C	CB53	BIT 2,E	CB55	BIT 2,L		
CB59	BIT 3,C	CB5B	BIT 3,E	CB5D	BIT 3,L		
CB61	BIT 4,C	CB63	BIT 4,E	CB65	BIT 4,L		
CB69	BIT 5,C	CB6B	BIT 5,E	CB6D	BIT 5,L		
CB71	BIT 6,C	CB73	BIT 6,E	CB75	BIT 6,L		
CB79	BIT 7,C	CB7B	BIT 7,E	CB7D	BIT 7,L		

146. BIT b,(HL) Efect:  $Z \leftarrow \overline{(HL)_b}$ ; se introduce complementul față de 1 al bitului b din locația de memorie cu adresa conținută în registrul dublu HL, în bitul indicator de 0, Z (din registrul F).

CB46	BIT 0,(HL)	CB5E	BIT 3,(HL)	CB76	BIT 6,(HL)
CB4E	BIT 1,(HL)	CB66	BIT 4,(HL)	CB7E	BIT 7,(HL)
CB56	BIT 2,(HL)	CB6E	BIT 5,(HL)		

147. BIT b,(IX+d) Efect:  $Z \leftarrow \overline{(IX+d)_b}$ ; similar cu instrucțiunea 146, dar locația de memorie are adresa IX+d.

DDCB d46	BIT 0,(IX+d)	DDCB d66	BIT 4,(IX+d)
DDCB d4E	BIT 1,(IX+d)	DDCB d6E	BIT 5,(IX+d)
DDCB d56	BIT 2,(IX+d)	DDCB d76	BIT 6,(IX+d)
DDCB d5E	BIT 3,(IX+d)	DDCB d7E	BIT 7,(IX+d)

148. BIT b,(IY+d) Efect:  $Z \leftarrow \overline{(IY+d)_b}$ ; similar cu instrucțiunea 146, dar locația de memorie are adresa IY+d.

FDCB d46	BIT 0,(IY+d)	FDCB d66	BIT 4,(IY+d)
FDCB d4E	BIT 1,(IY+d)	FDCB d6E	BIT 5,(IY+d)
FDCB d56	BIT 2,(IY+d)	FDCB d76	BIT 6,(IY+d)
FDCB d5E	BIT 3,(IY+d)	FDCB d7E	BIT 7,(IY+d)

149. RES b,r Efect:  $r_b \leftarrow 0$ ; bitul b al registrului r este șters.

CB80	RES 0,B	CB81	RES 0,C	CB82	RES 0,D	CB83	RES 0,E
CB88	RES 1,B	CB89	RES 1,C	CB8A	RES 1,D	CB8B	RES 1,E
CB90	RES 2,B	CB91	RES 2,C	CB92	RES 2,D	CB93	RES 2,E
CB98	RES 3,B	CB99	RES 3,C	CB9A	RES 3,D	CB9B	RES 3,E
CBA0	RES 4,B	CBA1	RES 4,C	CBA2	RES 4,D	CBA3	RES 4,E
CBA8	RES 5,B	CBA9	RES 5,C	CBAA	RES 5,D	CBAB	RES 5,E
CBB0	RES 6,B	CBB1	RES 6,C	CBB2	RES 6,D	CBB3	RES 6,E



CBB8	RES 7,B	CBB9	RES 7,C	CBBA	RES 7,D	CBBB	RES 7,E
CB84	RES 0,H	CB85	RES 0,L	CB87	RES 0,A		
CB8C	RES 1,H	CB8D	RES 1,L	CB8F	RES 1,A		
CB94	RES 2,H	CB95	RES 2,L	CB97	RES 2,A		
CB9C	RES 3,H	CB9D	RES 3,L	CB9F	RES 3,A		
CBA4	RES 4,H	CBA5	RES 4,L	CBA7	RES 4,A		
CBAC	RES 5,H	CBAD	RES 5,L	CBAF	RES 5,A		
CBB4	RES 6,H	CBB5	RES 6,L	CBB7	RES 6,A		
CBBC	RES 7,H	CBBD	RES 7,L	CBBF	RES 7,A		

150. RES b,(HL) Efect:  $(HL)_b \leftarrow 0$ ; bitul b al locației de memorie cu adresa conținută în registrul dublu HL este șters.

CB86	RES 0,(HL)	CB9E	RES 3,(HL)	CBB6	RES 6,(HL)
CB8E	RES 1,(HL)	CBA6	RES 4,(HL)	CBBE	RES 7,(HL)
CB96	RES 2,(HL)	CBAE	RES 5,(HL)		

151. RES b,(IX+d) Efect:  $(IX+d)_b \leftarrow 0$ ; similar cu instrucțiunea 150, dar locația de memorie are adresa IX+d.

DDCB d86	RES 0,(IX+d)	DDCB dA6	RES 4,(IX+d)
DDCB d8E	RES 1,(IX+d)	DDCB dAE	RES 5,(IX+d)
DDCB d96	RES 2,(IX+d)	DDCB dB6	RES 6,(IX+d)
DDCB d9E	RES 3,(IX+d)	DDCB dBE	RES 7,(IX+d)

152. RES b,(IY+d) Efect:  $(IY+d)_b \leftarrow 0$ ; similar cu instrucțiunea 150, dar locația de memorie are adresa IY+d.

FDCB d86	RES 0,(IY+d)	FDCB dA6	RES 4,(IY+d)
FDCB d8E	RES 1,(IY+d)	FDCB dAE	RES 5,(IY+d)
FDCB d96	RES 2,(IY+d)	FDCB dB6	RES 6,(IY+d)
FDCB d9E	RES 3,(IY+d)	FDCB dBE	RES 7,(IY+d)

153. SET b,r Efect:  $r_b \leftarrow 1$ ; bitul b din registrul r este înscris cu valoarea logică 1.

CBC0	SET 0,B	CBC1	SET 0,C	CBC2	SET 0,D	CBC3	SET 0,E
CBC8	SET 1,B	CBC9	SET 1,C	CBCA	SET 1,D	CBCB	SET 1,E
CBD0	SET 2,B	CBD1	SET 2,C	CBD2	SET 2,D	CBD3	SET 2,E
CBD8	SET 3,B	CBD9	SET 3,C	CBDA	SET 3,D	CBDB	SET 3,E
CBE0	SET 4,B	CBE1	SET 4,C	CBE2	SET 4,D	CBE3	SET 4,E
CBE8	SET 5,B	CBE9	SET 5,C	CBEA	SET 5,D	CBEB	SET 5,E
CBF0	SET 6,B	CBF1	SET 6,C	CBF2	SET 6,D	CBF3	SET 6,E
CBF8	SET 7,B	CBF9	SET 7,C	CBFA	SET 7,D	CBFB	SET 7,E
CBC4	SET 0,H	CBC5	SET 0,L	CBC7	SET 0,A		
CBCC	SET 1,H	CBCE	SET 1,L	CBCF	SET 1,A		
CBD4	SET 2,H	CBD5	SET 2,L	CBD7	SET 2,A		
CBDC	SET 3,H	CBDD	SET 3,L	CBDF	SET 3,A		
CBE4	SET 4,H	CBE5	SET 4,L	CBE7	SET 4,A		
CBEC	SET 5,H	CBED	SET 5,L	CBEF	SET 5,A		
CBF4	SET 6,H	CBF5	SET 6,L	CBF7	SET 6,A		
CBFC	SET 7,H	CBFD	SET 7,L	CBFF	SET 7,A		

154. SET b,(HL) Efect:  $(HL)_b \leftarrow 1$ ; bitul b al locației de memorie cu adresa conținută în registrul dublu HL este înscris cu valoarea logică 1.

CBC6	SET 0,(HL)	CBE6	SET 4,(HL)
CBCE	SET 1,(HL)	CBEE	SET 5,(HL)
CBD6	SET 2,(HL)	CBF6	SET 6,(HL)
CBDE	SET 3,(HL)	CBFE	SET 7,(HL)

155. SET b,(IX+d) Efect:  $(IX+d)_b \leftarrow 1$ ; similar cu instrucțiunea 154, dar locația de memorie are adresa IX+d.

DDCB dC6	SET 0,(IX+d)	DDCB dE6	SET 4,(IX+d)
DDCB dCE	SET 1,(IX+d)	DDCB dEE	SET 5,(IX+d)
DDCB dD6	SET 2,(IX+d)	DDCB dF6	SET 6,(IX+d)
DDCB dDE	SET 3,(IX+d)	DDCB dFE	SET 7,(IX+d)



156. SET b,(IY+d) Efect:  $(IY+d)_b \leftarrow 1$ ; similar cu instrucțiunea 154, dar locația de memorie are adresa IY+d.

FDCB dC6	SET 0,(IY+d)	FDCB dE6	SET 4,(IY+d)
FDCB dCE	SET 1,(IY+d)	FDCB dEE	SET 5,(IY+d)
FDCB dD6	SET 2,(IY+d)	FDCB dF6	SET 6,(IY+d)
FDCB dDE	SET 3,(IY+d)	FDCB dFE	SET 7,(IY+d)

### Grupul instrucțiunilor de legătură

Instrucțiunile acestui grup schimbă ordinea secvențială a programului prin salturi, apeluri de subrutină, reveniri din subrutină, necondiționat, sau în funcție de o condiție specificată.

Indicatorii de condiții sînt testați așa cum se descrie în subcapitolul 3.2, în care se precizează și care este efectul instrucțiunilor asupra acestor indicatori.

Nici o instrucțiune a acestui grup nu afectează indicatorii de condiții.

157. JP n n Efect:  $PC \leftarrow nn$ ; operandul nn este încărcat în perechea de registre C3 n n PC (numărător de program) pentru a indica adresa următoarei instrucțiuni care se va executa.

158. JP NZ,nn Efect: continuă, dacă  $Z=1$ ;  $PC \leftarrow nn$  dacă  $Z=0$ ; dacă  $Z=0$ , operandul nn este încărcat în perechea de registre PC (numărător de program) pentru a indica adresa următoarei instrucțiuni care se va executa; dacă  $Z=1$ , numărătorul de program este incrementat ca și în cazul instrucțiunilor care nu sînt de salt și programul continuă cu următoarea instrucțiune, secvențial; primul operand din codul obiect asamblat este octetul inferior al adresei de memorie nn.

159. JP Z,nn Efect: continuă dacă  $Z=0$ ;  $PC \leftarrow nn$  dacă  $Z=1$ ; efectul este similar cu al instrucțiunii 158.

160. JP NC, nn Efect: continuă dacă  $C=1$ ;  $PC \leftarrow nn$  dacă  $C=0$ ; efectul este similar cu al instrucțiunii 158.

161. JP C,nn Efect: continuă dacă  $C=0$ ;  $PC \leftarrow nn$ , dacă  $C=1$ ; efectul este similar cu al instrucțiunii 158.

162. JP PO,nn Efect: continuă dacă  $P/V=1$ ;  $PC \leftarrow nn$  dacă  $P/V=0$ ; efectul este similar cu al instrucțiunii 158.

163. JP PE,nn Efect: continuă dacă  $P/V=0$ ;  $PC \leftarrow nn$  dacă  $P/V=1$ ; efectul este similar cu al instrucțiunii 158.

164. JP P,nn Efect: continuă dacă  $S=1$ ;  $PC \leftarrow nn$ , dacă  $S=0$ ; efectul este similar cu al instrucțiunii 158.

165. JP M,nn Efect: continuă dacă  $S=0$ ;  $PC \leftarrow nn$ , dacă  $S=1$ ; efectul este similar cu al instrucțiunii 158.

166. JP (HL) Efect:  $PC \leftarrow HL$ ; numărătorul de program PC este încărcat cu conținutul perechii de registre HL pentru a indica adresa următoarei instrucțiuni care se va executa.

167. JP (IX) Efect:  $PC \leftarrow IX$ ; numărătorul de program PC este încărcat cu conținutul registrului index IX, pentru a indica adresa următoarei instrucțiuni care se va executa.

168. JP (IY) Efect:  $PC \leftarrow IY$ ; similar cu instrucțiunea 167, conținutul numărătorului de program fiind preluat din registrul index IY.

169. JR c Efect:  $PC \leftarrow PC + c$ ; valoarea deplasamentului c este adunată la conținutul numărătorului de program PC, iar următoarea instrucțiune este adusă din locația indicată de noul conținut al numărătorului de program; saltul se măsoară de la adresa octetului care indică codul operației și are lungimea în domeniul  $-126 \div 129$  octeți.

170. JR NZ,c Efect: continuă pentru  $Z=1$ ;  $PC \leftarrow PC + c$  pentru  $Z=0$ ; dacă bitul indicator de zero este 0, valoarea deplasamentului c este adunată



la conținutul numărătorului de program PC, iar următoarea instrucțiune este adusă din locația indicată de noul conținut al numărătorului de program PC; saltul se măsoară de la adresa octetului care indică codul operației și are lungimea în domeniul  $-126 \div 129$  octeți; dacă bitul indicator de zero este 1, următoarea instrucțiune se aduce din locația care urmează prezenta instrucțiune.

171. JR Z,e  
28 e-2      Efect: continuă pentru  $Z=0$ ;  $PC \leftarrow PC + e$  pentru  $Z=1$ ; efectul este similar cu al instrucțiunii 170.
172. JR NC,e  
30 e-2      Efect: continuă pentru  $C=1$ ;  $PC \leftarrow PC + e$  pentru  $C=0$ ; efectul este similar cu al instrucțiunii 170.
173. JR C,e  
38 e-2      Efect: continuă pentru  $C=0$ ;  $PC \leftarrow PC + e$  pentru  $C=1$ ; efectul este similar cu al instrucțiunii 170.
174. DJNZ e  
10 e-2      Efect:  $B \leftarrow B - 1$ ; continuă pentru  $B=0$ ;  $PC \leftarrow PC + e$  pentru  $B \neq 0$ ; registrul B este decrementat și dacă rezultatul este nenul valoarea deplasamentului e este adunată la conținutul numărătorului de program PC; următoarea instrucțiune este adusă din locația de memorie indicată de noul conținut al numărătorului de program PC; saltul se măsoară de la adresa octetului care indică codul operației și are lungimea în domeniul  $-126 \div 129$  octeți; dacă după decrementare, conținutul lui B este 0, următoarea instrucțiune se aduce din locația care urmează prezenta instrucțiune.
175. CALL n n      Efect:  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow nn$ ; se depune conținutul  
CD n n      curent al numărătorului de program PC pe vârful stivei și operanzii nn sînt încărcăți în numărătorul de program, pentru a indica locația din care se va aduce codul operației primei instrucțiuni a unei subrutine (la sfîrșitul subrutinei poate fi utilizată o instrucțiune RET pentru revenirea la programul principal, încărcînd cei doi octeți depuși pe vârful stivei, în numărătorul de program); operația de depunere pe stivă are loc decrementînd la început conținutul curent al indicatorului de stivă, SP, încărcînd octetul de ordin superior din PC în locația de memorie indicată acum de SP, decrementînd SP din nou și încărcînd octetul de ordin inferior în locația de memorie indicată de SP; deoarece instrucțiunea CALL, nn are trei octeți, numărătorul de program PC este incrementat de 3 ori înainte de a fi depus pe stivă.
176. CALL NZ,nn      Efect: continuă, dacă  $Z=1$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow$   
C4 n n      nn, dacă  $Z=0$ ; dacă  $Z=0$ , are efect similar cu instrucțiunea CALL nn; dacă  $Z=1$ , numărătorul de program PC este incrementat secvențial.
177. CALL Z,nn      Efect: continuă, dacă  $Z=0$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \rightarrow PC_L$ ,  $PC \leftarrow nn$ .  
CC n n      dacă  $Z=1$ ; dacă  $Z=1$ , are efect similar cu instrucțiunea CALL nn; dacă  $Z=0$ , numărătorul de program PC este incrementat secvențial.
178. CALL NC,n n      Efect: continuă, dacă  $C=1$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow$   
D4 n n       $\leftarrow nn$ , dacă  $C=0$ ; dacă  $C=0$ , are efect similar cu instrucțiunea CALL nn; dacă  $C=1$ , numărătorul de program este incrementat în mod secvențial.
179. CALL C,nn      Efect: continuă dacă  $C=0$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow nn$ .  
DC n n      dacă  $C=1$ ; dacă  $C=1$ , are efect similar cu instrucțiunea CALL nn; dacă  $C=0$ , numărătorul de program este incrementat în mod secvențial.
180. CALL PO,nn      Efect: continuă, dacă  $P/V=1$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  
E4 n n       $PC \leftarrow nn$ , dacă  $P/V=0$ ; dacă  $P/V=0$ , are efect similar cu instrucțiunea CALL, nn; dacă  $P/V=1$ , numărătorul de program este incrementat secvențial.



181. CALL PE,nn Efect: continuă, dacă  $P/V=0$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow nn$ , dacă  $P/V=1$ ; dacă  $P/V=1$ , are efect similar cu instrucțiunea CALL nn; dacă  $P/V=0$ , numărătorul de program este incrementat în mod secvențial.
182. CALL P,nn Efect: continuă, dacă  $S=1$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow nn$ , dacă  $S=0$ ; dacă  $S=0$ , are efect similar cu instrucțiunea CALL nn; dacă  $S=1$ , numărătorul de program este incrementat secvențial.
183. CALL M,nn Efect: continuă, dacă  $S=0$ ;  $(SP-1) \leftarrow PC_H$ ,  $(SP-2) \leftarrow PC_L$ ,  $PC \leftarrow nn$ , dacă  $S=1$ ; dacă  $S=1$ , are efect similar cu instrucțiunea CALL nn; dacă  $S=0$ , numărătorul de program este incrementat în mod secvențial.
184. RET C9 Efect:  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ; se revine în programul principal preluând conținutul anterior al numărătorului de program de pe vârful stivei, unde a fost depus de instrucțiunea CALL; se încarcă în octetul de ordin inferior al lui PC conținutul locației de memorie indicată de SP, se incrementează SP, se încarcă în octetul de ordin superior al lui PC conținutul locației de memorie indicată acum de SP, iar SP este incrementat a doua oară; în timpul următorului ciclu de mașină, unitatea centrală va aduce următorul cod de operație din locația indicată acum de PC.
185. RET NZ C0 Efect: continuă, dacă  $Z=1$ ;  $PC_L \leftarrow (SP)$ ;  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $Z=0$ ; dacă  $Z=0$ , are efect similar cu instrucțiunea RET; dacă  $Z=1$ , numărătorul de program este incrementat secvențial.
186. RET Z C8 Efect: continuă, dacă  $Z=0$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $Z=1$ ; dacă  $Z=1$ , are efect similar cu instrucțiunea RET; dacă  $Z=0$ , numărătorul de program este incrementat secvențial.
187. RET NC D0 Efect: continuă, dacă  $C=1$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $C=0$ ; dacă  $C=0$ , are efect similar cu instrucțiunea RET; dacă  $C=1$ , numărătorul de program este incrementat secvențial.
188. RET C D8 Efect: continuă, dacă  $C=0$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $C=1$ ; dacă  $C=1$ , are efect similar cu instrucțiunea RET; dacă  $C=0$ , numărătorul de program este incrementat secvențial.
189. RET PO E0 Efect: continuă, dacă  $P/V=1$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $P/V=0$ ; dacă  $P/V=0$ , are efect similar cu instrucțiunea RET; dacă  $P/V=1$ , numărătorul de program este incrementat în mod obișnuit.
190. RET PE E8 Efect: continuă, dacă  $P/V=0$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $P/V=1$ ; dacă  $P/V=1$ , are efect similar cu instrucțiunea RET; dacă  $P/V=0$ , numărătorul de program este incrementat secvențial.
191. RET P F0 Efect: continuă, dacă  $S=1$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $S=0$ ; dacă  $S=0$ , are efect similar cu instrucțiunea RET; dacă  $S=1$ , numărătorul de program este incrementat secvențial.
192. RET M F8 Efect: continuă, dacă  $S=0$ ;  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ , dacă  $S=1$ ; dacă  $S=1$ , are efect similar cu instrucțiunea RET; dacă  $S=0$ , numărătorul de program este incrementat secvențial.
193. RETI ED4D Efect:  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP+1$ ; instrucțiunea de revenire din întrerupere este utilizată la sfârșitul unei subrutine



de deservire a unei întreruperi pentru a reface conținutul număratorului de program PC (analog cu instrucțiunea RET) și pentru a semnaliza unui dispozitiv de I/E că rutina de întrerupere a fost executată; instrucțiunea RETI permite suprapunerea subrutinelor de întrerupere, dispozitivele cu prioritate mai mare suspendând deservirea rutinelor de prioritate mai mică; starea bistabilului IFF2 este copiată în bistabilul IFF1.

194. RETN  
ED45

Efect:  $PC_L \leftarrow (SP)$ ,  $SP \leftarrow SP + 1$ ,  $PC_H \leftarrow (SP)$ ,  $SP \leftarrow SP + 1$ ; instrucțiunea de revenire din întrerupere nemascabilă este utilizată la sfârșitul unei rutine de deservire a unei cereri  $\overline{NMI}$  și execută o revenire necondiționată, în mod similar cu cea din instrucțiunea RET; controlul revine la programul principal, iar în următorul ciclu de mașină unitatea centrală aduce codul operației unei instrucțiuni din locația indicată de PC; starea lui IFF2 este recopiată în IFF1, revenind la starea pe care o avea înainte de cererea de întrerupere  $\overline{NMI}$ , la care programul a continuat de la adresa 0066H.

195. RST p

Efect:  $(SP - 1) \leftarrow PC_H$ ,  $(SP - 2) \leftarrow PC_L$ ,  $PC_H \leftarrow 00$ ,  $PC_L \leftarrow p$ ; instrucțiunea de restart salvează conținutul curent al număratorului de program pe stivă, iar adresa p este încărcată în număratorului de program; programul continuă cu instrucțiunea de la această adresă, ceea ce constituie un apel la adresa p; octetul al doilea al instrucțiunii este încărcat în  $PC_H$  iar  $PC_L$  va conține 00H.

C7 RST 00H D7 RST 10H E7 RST 20H F7 RST 30H  
CF RST 08H DF RST 18H EF RST 28H FF RST 38H

Grupul instrucțiunilor de lucru cu stiva, de intrare/ieșire și de control.

Instrucțiunile acestui grup realizează lucrul cu stiva, operații de intrare și ieșire, modifică indicatorii de condiții interni sau fixează modul de tratare a întreruperilor.

196. EX (SP), HL Efect:  $H \leftrightarrow (SP + 1)$ ,  $L \leftrightarrow (SP)$ ; octetul inferior conținut în perechea E3

de registre HL este schimbat cu conținutul locației de memorie specificată de conținutul indicatorului de stivă SP, iar octetul superior al perechii HL este schimbat cu conținutul locației de memorie cu adresa  $SP + 1$ .

197. EX (SP), IX Efect:  $IX_H \leftrightarrow (SP + 1)$ ,  $IX_L \leftrightarrow (SP)$ ; efect similar cu al instrucțiunii DDE3 196, dar vârful stivei este schimbat cu conținutul registrului dublu IX.

198. EX (SP), IY Efect:  $IY_H \leftrightarrow (SP + 1)$ ,  $IY_L \leftrightarrow (SP)$ , efect similar cu al instrucțiunii FDE3 196, dar vârful stivei este schimbat cu conținutul registrului dublu IY.

199. PUSH qq Efect:  $(SP - 2) \leftarrow qq_L$ ,  $(SP - 1) \leftarrow qq_H$ ; conținutul perechii de registre qq este salvat în memoria externă LIFO (ultimul intrat, primul ieșit); se decrementează la început SP și se transferă octetul superior al perechii qq în locația de memorie cu adresa SP; se decrementează din nou SP și se transferă octetul inferior al perechii qq în locația de memorie cu adresa conținută acum în SP.

C5 PUSH BC D5 PUSH DE E5 PUSH HL F5 PUSH AF

200. PUSH IX Efect:  $(SP - 2) \leftarrow IX_L$ ,  $(SP - 1) \leftarrow IX_H$ ; similar cu instrucțiunea PUSH DDE5 qq, dar se salvează conținutul lui IX pe stivă.

201. PUSH IY Efect:  $(SP - 2) \leftarrow IY_L$ ,  $(SP - 1) \leftarrow IY_H$ ; similar cu instrucțiunea PUSH FDE5 qq, dar se salvează pe stivă conținutul lui IY.

202. POP qq Efect:  $qq_H \leftarrow (SP + 1)$ ;  $qq_L \leftarrow (SP)$ ; cei doi octeți superiori din vârful memoriei externe LIFO (ultimul intrat — primul ieșit) care constituie stiva, sînt transferați în perechea de registre qq; indicatorul de stivă conține adresa de 16 biți a locației care constituie vârful stivei; instrucțiunea încarcă în octetul inferior al perechii qq conținutul locației de memorie cu adresa în registrul dublu SP; indicatorul de stivă este în continuare incrementat, iar conținutul locației de memo-



rie cu adresa conținută acum în SP este transferat în octetul superior al perechii qq. În final, SP este incrementat din nou.

CI POP BC DI POP DE EI POP HL FI POP AF

203. POP IX DDEI Efect:  $IX_H \leftarrow (SP+1)$ ,  $IX_L \leftarrow (SP)$ ; similar cu instrucțiunea POP qq, dar cei 2 octeți preluați de pe vârful stivei sînt transferați în registrul dublu IX.
204. POP IV FDEI Efect:  $IV_H \leftarrow (SP+1)$ ,  $IV_L \leftarrow (SP)$ ; similar cu instrucțiunea POP qq, dar cei 2 octeți preluați de pe vârful stivei sînt transferați în registrul dublu IV.
205. IN A,(n) DB n Efect:  $A \leftarrow (n)$ ; operandul n este plasat pe jumătatea inferioară a magistralei de adrese ( $A_0-A_7$ ) pentru a selecta un dispozitiv de I/E la unul dintre cele 256 de porturi posibile; conținutul acumulatorului apare în același timp pe jumătatea superioară a magistralei de adrese ( $A_8-A_{15}$ ); în continuare, un octet de la portul selectat este plasat pe magistrala de date și este înscris în unitatea centrală, în acumulatorul A.
206. IN r,(C) Efect:  $r \leftarrow (C)$ ; conținutul registrului C este plasat pe liniile de adresă  $A_0-A_7$ , pentru a selecta un dispozitiv de I/E la unul dintre cele 256 de porturi posibile; conținutul registrului B este plasat în același timp pe liniile de adresă  $A_8-A_{15}$ ; în continuare, un octet de la portul selectat este plasat pe magistrala de date și este înscris în unitatea centrală, în registrul r.
- ED40 IN B,(C) ED50 IN D,(C) ED60 IN H,(C) ED78 IN A,(C)  
ED48 IN C,(C) ED58 IN E,(C) ED68 IN L,(C) ED70 IN F,(C)
207. IND EDAA Efect:  $(HL) \leftarrow (C)$ ,  $B \leftarrow B-1$ ,  $HL \leftarrow HL-1$ ; conținutul registrului C este plasat pe liniile de adresă  $A_0-A_7$  pentru a selecta un dispozitiv de I/E, la unul dintre cele 256 de porturi posibile; registrul B poate fi folosit ca numărator de octeți și conținutul lui este plasat pe liniile de adresă  $A_8-A_{15}$ , în același timp; în continuare, un octet de la portul selectat este plasat pe magistrala de date și este înscris în unitatea centrală; conținutul registrului dublu HL este plasat pe magistrala de adrese și octetul citit de la portul selectat este înscris în locația de memorie cu adresa în HL; în final, număratorul de octeți B și registrul dublu HL sînt decrementate.
208. INDR EDBA Efect:  $(HL) \leftarrow (C)$ ,  $B \leftarrow B-1$ ,  $HL \leftarrow HL-1$ , pînă cînd se obține  $B=0$ ; conținutul registrului C este plasat pe liniile de adresă  $A_0-A_7$ , pentru a selecta un dispozitiv de I/E; registrul B este utilizat ca numărator de octeți, iar conținutul lui este plasat în același timp pe liniile de adresă  $A_8-A_{15}$ ; în continuare, un octet de la portul selectat este plasat pe magistrala de date și este înscris în unitatea centrală; conținutul registrului dublu HL este plasat pe magistrala de adrese și octetul citit de la portul selectat este înscris în locația de memorie cu adresa conținută în registrul HL; registrul dublu HL și număratorul de octeți B sînt decrementate; dacă prin decrementare rezultă  $B=0$ , instrucțiunea este terminată; dacă  $B \neq 0$ , număratorul de program PC este decrementat de două ori și instrucțiunea se repetă; de notat că  $B=0$  înaintea execuției instrucțiunii determină citirea a 256 de octeți de date de la portul selectat; întreruperile sînt recunoscute și se execută două cicluri de refreșare după fiecare transfer de date.
209. INI EDA2 Efect:  $(HL) \leftarrow (C)$ ,  $B \leftarrow B-1$ ,  $HL \leftarrow HL+1$ ; efectul este similar cu al instrucțiunii IND, dar registrul dublu HL este incrementat la sfîrșitul executării instrucțiunii.



210. INIR EDB2 Efect:  $(HL) \leftarrow (C)$ ,  $B \leftarrow B - 1$ ,  $HL \leftarrow HL + 1$ , pînă cînd se obține  $B = 0$ ; efectul este similar cu al instrucțiunii INDR, dar conținutul registrului dublu HL este incrementat după fiecare transfer de la portul adresat în memorie.
211. OUT (n),A D3 n Efect:  $(n) \leftarrow A$ ; operandul n este plasat pe liniile de adresă  $A_0 - A_7$  pentru a selecta un dispozitiv de I/E, la unul dintre cele 256 de porturi posibile; conținutul acumulatorului A apare în același timp pe liniile de adresă  $A_8 - A_{15}$ ; în continuare, octetul conținut în acumulator este plasat pe magistrala de date și este înscris în dispozitivul periferic selectat.
212. OUT (C),r Efect:  $(C) \leftarrow r$ ; conținutul registrului C este plasat pe liniile de adresă  $A_0 - A_7$  pentru a selecta un dispozitiv de I/E; conținutul registrului B este plasat pe liniile de adresă  $A_8 - A_{15}$ , în același timp; în continuare, octetul conținut în registrul r este plasat pe magistrala de date și este înscris în dispozitivul periferic selectat.
- ED41 OUT (C),B ED51 OUT (C),D ED61 OUT (C),H ED79 OUT (C),A  
ED49 OUT (C),C ED59 OUT (C),E ED69 OUT (C),L
213. OUTD EDAB Efect:  $(C) \leftarrow (HL)$ ,  $B \leftarrow B - 1$ ,  $HL \leftarrow HL - 1$ ; instrucțiune de ieșire similară cu instrucțiunea de intrare IND.
214. OTDR EDBB Efect:  $(C) \leftarrow (HL)$ ,  $B \leftarrow B - 1$ ,  $HL \leftarrow HL - 1$ , pînă cînd se obține  $B = 0$ ; instrucțiune de ieșire cu efect similar cu al instrucțiunii de intrare INDR.
215. OUTI EDA3 Efect:  $(C) \leftarrow (HL)$ ,  $B \leftarrow B - 1$ ,  $HL \leftarrow HL + 1$ ; instrucțiunea de ieșire cu efect similar cu al instrucțiunii de intrare INI.
216. OTIR EDB3 Efect:  $(C) \leftarrow (HL)$ ,  $B \leftarrow B - 1$ ,  $HL \leftarrow HL + 1$ , pînă cînd se obține  $B = 0$ ; instrucțiune de ieșire cu efect similar cu al instrucțiunii INIR.
217. EI FB Efect:  $IFF \leftarrow 1$ ; înscrie valoarea 1 în bistabilele de acceptare a întreruperilor IFF1 și IFF2, determinînd acceptarea întreruperilor mascabile; nu sînt acceptate aceste întreruperi în timpul executării instrucțiunii EI.
218. DI F3 Efect:  $IFF \leftarrow 0$ ; se șterg bistabilele de acceptare a întreruperilor, IFF1 și IFF2; după executarea instrucțiunii nu se mai acceptă întreruperi mascabile; acestea nu sînt acceptate nici în timpul executării instrucțiunii DI.
219. IM 0 ED46 Efect: se trece în modul 0 de tratare a întreruperilor; dispozitivul care a cerut întreruperea poate insera orice instrucțiune pe magistrala de date, permițînd unității centrale să o execute.
220. IM 1 ED56 Efect: se trece în modul 1 de tratare a întreruperilor, în care microprocesorul răspunde la o cerere de întrerupere prin executarea unei instrucțiuni de restart la locația 0038H.
221. IM 2 ED5E Efect: se trece în modul 2 de tratare a întreruperilor, în care se permite un apel indirect la orice adresă din memorie; în acest mod unitatea centrală formează o adresă de 16 biți: cei 8 biți superiori sînt conținutul registrului vectorului de întreruperi I, iar cei 8 biți inferiori sînt furnizați de dispozitivul care a cerut întreruperea; la această adresă se află adresa subrutinei de întrerupere.
222. HALT 76 Efect: se suspendă activitatea unității centrale pînă la primirea unei cereri de întrerupere sau a unui semnal de inițializare (reset); în timpul stării „HALT”, procesorul va executa instrucțiuni NOP pentru a menține logica de refreșare a memoriei.
223. NOP 00 Efect: unitatea centrală nu execută nici o operație în timpul acestei instrucțiuni.

Tabelul 3.1 conține, sintetic, numărul de stări necesare executării fiecărei instrucțiuni a microprocesorului, ceea ce permite calculul timpului de execuție al unui



program, cunoscând durata unei stări. De notat că durata următoarelor instrucțiuni depinde de îndeplinirea unei condiții:

LDDR, LDIR au durata de 21 de stări dacă  $BC \neq 0$  și de 16 stări dacă  $BC = 0$

CPDR, CPDR au durata de 21 de stări dacă  $BC \neq 0$  și  $A \neq (HL)$  și de 16 stări dacă  $BC = 0$  sau  $A = (HL)$

JR C,e; JR NC,e; JR Z,e; JR NZ,e au durata de 12 stări dacă este îndeplinită condiția de salt și de 7 stări în caz contrar.

DJNZ e are durata de 13 stări dacă  $B \neq 0$  și 8 stări pentru  $B = 0$ .

CALL cc,nn au durata de 17 stări dacă este îndeplinită condiția de apel și de 10 stări în caz contrar

RET cc au durata de 11 stări dacă este îndeplinită condiția de salt și de 5 stări în caz contrar

INIR, INDR, OTIR, OTDR au durata de 21 de stări pentru  $B \neq 0$  și de 16 stări pentru  $B = 0$ .

Trebuie remarcat faptul că bitul indicator de transport C, din registrul F, a fost notat în unele cazuri cu CY, pentru a evita confuzia cu registrul C al microprocesorului.

Numărul de stări corespunzător fiecărei instrucțiuni a microprocesorului Z 80

Tabelul 3.1

Grup transfer de 8 biți	Grup schimb, transfer, căutare de bloc	OR n	7	INC IX	10	BIT b,(IX+d)	20	
LD r,r	4	OR (HL)	7	INC IY	10	BIT b,(IY+d)	20	
LD r,n	7	OR (IX+d)	19	DEC ss	6	SET b,r	8	
LD n,(HL)	7	OR (IY+d)	19	DEC IX	10	SET b,(HL)	15	
LD r,(IX+d)	19	XOR r	4	DEC IY	10	SET b,(IX+d)	23	
LD r,(IY+d)	19	XOR n	7	Grup rotiri/deplasări		SET b,(IY+d)	23	
LD (HL), r	7	XOR (HL)	7		RLCA	4	RES b,r	8
LD (IX+d),r	19	XOR (IX+d)	19		RLA	4	RES b,(HL)	15
LD (IY+d),r	19	XOR (IY+d)	19		RRCA	4	RES b,(IX+d)	23
LD (HL),n	10	CP r	4	RRA	4	RES b,(IY+d)	23	
LD (IX+d),n	19	CP n	7	Grup de salt				
LD (IY+d),n	19	CP (HL)	7		JP nn	10		
LD A,(BC)	7	CP (IX+d)	19		JP cc,nn	10		
LD A,(DE)	7	CP (IY+d)	19		JR e	12		
LD A,(nn)	13	INC r	4		JR C,e	7/12		
LD (BC),A	7	INC (HL)	11		JR NC,e	7/12		
LD (DE),A	7	INC (IX+d)	23		JR Z,e	7/12		
LD (nn),A	13	INC (IY+d)	23		JR NZ,e	7/12		
LD A,I	9	DEC r	4		JP (HL)	4		
LD A,R	9	DEC (HL)	11		JP (IX)	8		
LD I,A	9	DEC (IX+d)	23		JP (IY)	8		
LD R,A	9	DEC (IY+d)	23		DJNZ e	8/13		
Grup transfer de 16 biți	Grup aritmetic/logic de 8 biți	Grup aritmetică generală/control CPU			RRC (IX+d)	23	Grup de apel/revenire	
					RRC (IY+d)	23		
					RR r	8		
					RR (HL)	15		
			DAA	4	RR (IX+d)	23	CALL nn	17
			CPL	4	RR (IY+d)	23	CALL cc,nn	10/17
			NEG	8	SLA r	8	RET	10
			CCF	4	SLA (HL)	15	RET cc	5/11
			SCF	4	SLA (IX+d)	23	RETI	14
			NOP	4	SLA (IY+d)	23	RETN	14
		HALT	4	SRA r	8	RST p	11	
		DI	4	SRA (HL)	15	Grup operații intrare/ieșire		
		EI	4	SRA (IX+d)	23		IN A,(n)	11
		IMO	8	SRA (IY+d)	23		IN r,(C)	12
		IM1	8	SRL r	8		INI	16
		IM2	8	SRL (HL)	15		INIR	21/16
		Grup aritmetic de 16 biți	SRL (IX+d)	23	IND		16	
			SRL (IY+d)	23	INDR		21/16	
			RLD	18	OUT (n),A		11	
			RRD	18	OUT (C),r		12	
			Grup înscriere/ștergere/test de bit		OUTI		16	
		ADD HL,ss		11	OTIR	21/16		
		ADC HL,ss		15	OUTD	16		
		SBC HL,ss		15	OTDR	21/16		
		ADD IX,pp	15					
		ADD IY,rr	15					
	INC ss	6						



### 3.2. INDICATORI DE CONDIȚII ȘI OPERAȚII ARITMETICE

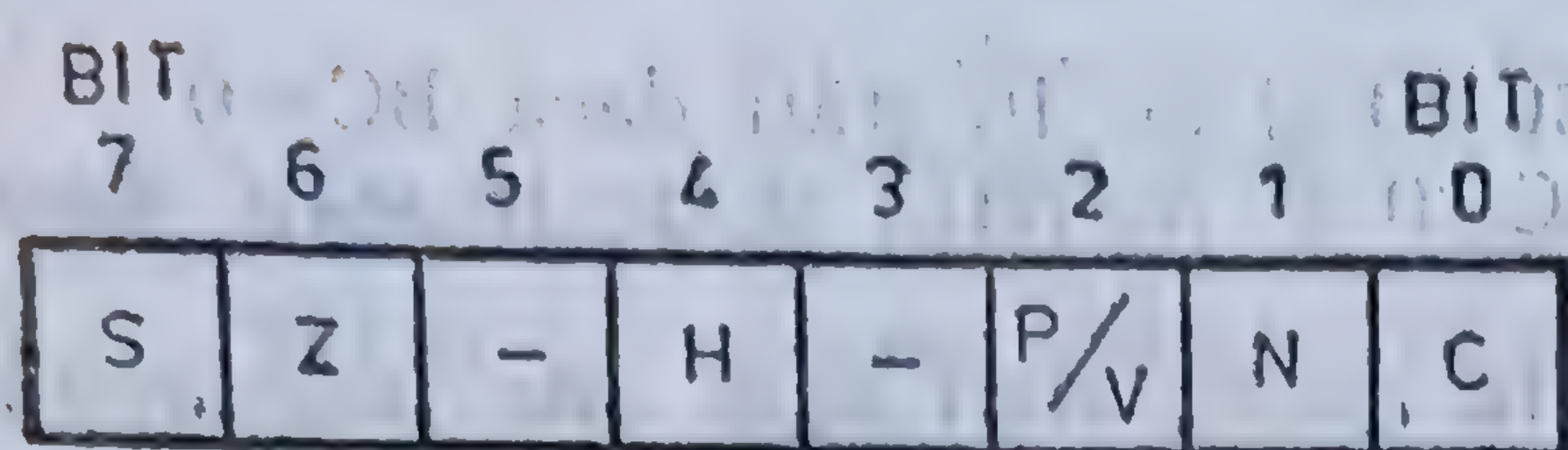


Fig. 3.10 Registrul indicatorilor de condiții.

Indicatorii de condiții sînt bistabile grupate în interiorul unității centrale, astfel încît să simplifice salvarea conținutului lor în memorie sau refacerea acestui conținut. Formatul registrului indicatorilor de condiții este cel din figura 3.10.

#### 1. Z, bitul indicator de zero

Bitul indicator de zero este înscris dacă rezultatul anumitor instrucțiuni este zero și este șters în caz contrar, conform tab. 3.2.

Acțiunile asupra indicatorului de zero (Z)

Tabelul 3..

Grup	Instrucțiuni	Acțiunea
Încărcare de 8 biți	LD A,I; LD A,R	Z=1 dacă I/R=0, altfel Z=0
Căutare	CPI; CPIR; CPD; CPDR	Z=1 dacă A(HL), altfel Z=0
Aritmetic de 8 biți	ADD A,s; SUB s; ADC A,s; INC s; DEC s; SBC A,s	
Logic de 8 biți	OR s; AND s; XOR s; CP s	
Aritmetic general	DAA; NEG	Z=1 dacă rezultatul este zero, altfel Z=0
Aritmetic de 16 biți	ADC HL,ss; SBC HL,ss	
Rotire și deplasare	RLC s; RL s; RRC s; RR s; SLA s; SRA s; SRL s; RLD; RRD	
Test de bit	BIT b,s	Z=1 dacă s <sub>b</sub> =0, altfel Z=0
Intrare/ieșire	IN r,(C)	Z=1 dacă data de intrare =0, altfel Z=0
	OUTI; OUTD; INI; IND	Z=1 dacă B-1=0, altfel Z=0
	OTIR; OTDR; INIR; INDR	Z=1

Bitul Z este afectat în special de operațiile aritmetice, logice și de deplasare. Încărcările și depunerile nu au efect asupra lui, cu excepția instrucțiunilor din tabelul de mai sus. Instrucțiunile de căutare îl afectează, conținând comparări și scăderi, ca și instrucțiunile de test de bit, care conțin un SI logic.

Este important faptul că odată fixat la 1 sau la zero de una din instrucțiunile de mai sus, bitul Z nu se mai schimbă pînă la întîlnirea unei alte instrucțiuni din același grup. Astfel, o instrucțiune de salt condiționat de bitul Z nu trebuie plasată imediat după instrucțiunea care a poziționat bitul Z, dacă alte instrucțiuni care afectează bitul Z nu o preced.

Bitul Z este testat într-o varietate de situații, cele mai obișnuite fiind: egalitatea a doi operanzi după o comparare; incrementarea sau decrementarea unui registru pînă la 0; testarea unui bit egal cu 0; rezultat 0 al unei deplasări; rezultat 0 al unui SI logic sau al unei operații aritmetice.

#### 2. S, bitul indicator de semn

Bitul S este înscris dacă rezultatul execuției anumitor instrucțiuni este negativ și este șters dacă rezultatul este pozitiv. Bitul S reflectă adevăratul semn al rezultatului, deoarece în aritmetica complementului față de 2, cantitățile pozitive au bitul 7 egal cu zero, iar cele negative, cu 1. Bitul S este afectat de instrucțiunile din tabelul 3.3.



Acțiunile asupra indicatorului de semn (S)

Tabelul 3.3.

Grup	Instrucțiuni	Acțiunea
Încărcare de 8 biți	LD A,I; LD A,R	S=1 dacă I/R este negativ, altfel S=0
Căutare	CPI; CPIR; CPD; CPDR	
Aritmetic de 8 biți	ADD A,s; ADC A,s; SUB s; SBC A,s; INC s; DEC s;	S=1 dacă rezultatul este negativ, altfel S=0
Logic de 8 biți	AND s; OR s; XOR s; CP s	
Aritmetic general	DAA NEG	S=1 dacă A <sub>7</sub> =1 altfel S=0
Aritmetic de 16 biți	ADC HL,ss; SBC HL,ss	S=1 dacă rezultatul este negativ, altfel S=0
Rotire și deplasare	RLC s; RL s; RRC s; RR s; SLA s; SRA s; SRL s RLD; RRD	S=1 dacă A este negativ după deplasare, altfel S=0
Test de bit	BIT b,s	Necunoscut S
Intrare/ieșire	IN r,(C) INI; INIR; IND; INDR; OUTI; OTIR; OUTD; OTDR	S=1 dacă data de intrare este negativă, altfel S=0 Necunoscut S

Modul de acțiune asupra bitului S este asemănător cu cel asupra bitului Z. Bitul S este afectat de operații aritmetice, logice sau de deplasare, incluzând comparațiile din grupul de căutare. Există instrucțiuni care afectează bitul S, dar starea lui este necunoscută. Testarea bitului S în cazul unor salturi condiționate se face asemănător cu testarea bitului Z. Situațiile în care se testează bitul S sînt în special: compararea a doi operanzi (mai mare, mai mic decît); incrementarea sau decrementarea unui registru peste valoarea zero; deplasarea unui 1 sau 0 în bi-stabilul S.

### 3. C, bitul indicator de transport

Instrucțiunile care afectează acest bit sînt în special cele aritmetice și de deplasare; cele logice îl șterg. Tabelul 3.4 indică toate situațiile în care este afectat bitul C.

Acțiunile asupra indicatorului de transport (C)

Tabelul 3.4.

Grup	Instrucțiuni	Acțiunea
Aritmetic de 8 biți	ADD A,s; ADC A,s;	C=1 la transport din bitul 7, altfel C=0
	SUB s; SBC A,s	C=1 dacă există împrumut, altfel C=0
Logic de 8 biți	AND s; OR s; XOR s	C=0
	CP s	C=1 dacă există împrumut, altfel C=0
Aritmetic general	DAA	C=1 la transport BCD, altfel C=0
	NEG	C=1 dacă A era diferit de 00H înainte, altfel C=0
	CCF	C=1 dacă C era 0 înainte, altfel C=0
	SCF	C=1
Aritmetic de 16 biți	ADD HL,ss; ADC HL,ss	C=1 la transport din bitul 15, altfel C=0
	SBC HL,ss	C=1 dacă există împrumut, altfel C=0
	ADD IX,pp; ADD IY,rr	C=1 la transport din bitul 15, altfel C=0



Grup	Instrucțiuni	Acțiunea
Rotire și deplasare	RLCA; RLA	$A_7 \rightarrow C$
	RRCA; RRA	$A_0 \rightarrow C$
	RLC s; RL s; SLA s	Bit 7 din operand $\rightarrow C$
	RRC s; RR s; SRA s; SRL s	Bit 0 din operand $\rightarrow C$

Bitul indicator de transport este utilizat în special pentru a testa rezultatul comparării a 2 operanzi, rezultatul unei operații de deplasare, sau pentru a realiza o operații aritmetice în precizie multiplă. Când se testează bitul de transport după compararea a 2 operanzi fără semn, bitul C va fi 1 dacă în compararea OP1:OP2 (efectuată cu ajutorul scăderii OP1-OP2), avem  $OP1 \geq OP2$ . Comparările pot fi făcute și prin testarea bitului de semn.

În instrucțiunile de deplasare, bitul C ia valoarea unui bit din operandul deplasat, ceea ce permite testarea și eventual saltul condiționat în funcție de valoarea unui bit al operandului. În adunările sau scăderile cu precizie multiplă, bitul C este fixat de cel mai semnificativ bit al rezultatului. În cazul adunării, prima este de tip ADD, iar următoarele de tip ADC (cu transport) pentru a aduna transportul de la adunarea anterioară.

Un exemplu de adunare și unul de scădere sînt date în continuare.

Adunare în precizie dublă:

+7287	Octet Sup.	Octet Inf.
+24572	00011100	01110111
+31859	01011111	11111100
	C=1 ←	01110011
	01111100	

Scădere în precizie dublă:

+426	00001010	00001010
-(+8193)	-00100000	00000001
+426	00001010	00001010
-8193	11011111	11111111
	C=1 ←	00001001
	11101010	

#### 4. P/V, bitul indicator de paritate/depășire

Bitul indicator de paritate/depășire are rol dublu. În cazul parității, este înscris pentru a reprezenta paritatea pară a rezultatului operației, ceea ce apare în suma celor 8 biți ai rezultatului este pară. Dacă suma este impară, bitul P/V este șters. De exemplu dacă rezultatul este 00101011, vom avea  $P/V=1$ , iar dacă este 00111110, vom avea  $P/V=0$ .

Cînd bitul P/V este utilizat pentru a indica o depășire, bitul este înscris cînd aceasta apare în urma unei operații aritmetice; dacă la adunarea a două numere de același semn, semnul rezultatului se schimbă, arătînd că rezultatul este prea



mare pentru a fi conținut în 8(16) biți, vom avea  $P/V=1$ . Cîteva exemple sînt prezentate în continuare.

+127 D	01111111	-125 D	10000011	+32 D	00100000
+64 D	01000000	-126 D	10000010	+32 D	00100000
+191 D	10111111	-251 D	00000101	+64 D	01000000
↓		↓		↓	
P/V=1 (depășire, rezultatul depășește 8 biți)		P/V=1 (depășire, rezultatul depășește 8 biți)		P/V=0 (nu există depășire)	

Instrucțiunile care afectează bitul indicator de paritate/depășire sînt cele din tabelul 3.5.

Acțiunile asupra indicatorului de paritate/depășire (P/V)

Tabelul 3.5.

Grup	Instrucțiuni	Acțiunea
Încărcare 8 biți	LD A,I; LD A,R	IFF2→P/V
Transfer de bloc și căutare	LDI; LDD; CPI; CPIR; CPD; CPDR	P/V=1 dacă $BC-1 \neq 0$ , altfel P/V=0
	LDIR; LDDR	P/V=0
Aritmetic de 8 biți	ADD A,s; ADC A,s; SUB s; SBC A,s	P/V=1 la depășire, altfel P/V=0
	INC s	P/V=1 pentru operand 7FH înainte de incrementare, altfel P/V=0
	DEC s	P/V=1 pentru operand 80H înainte de decrementare, altfel P/V=0
Logic de 8 biți	AND s; OR s; XOR s	P/V=1 la paritate pară, altfel P/V=0
	CP s	P/V=1 la depășire, altfel P/V=0
Aritmetic general	DAA	P/V=1 dacă A este par, altfel P/V=0
	NEG	P/V=1 pentru A=80H înainte de negare, altfel P/V=0
Aritmetic de 16 biți	ADC HL, ss; SBC HL, ss	P/V=1 la depășire, altfel P/V=0
Rotire și deplasare	RLC s; RL s; RRC s; RR s; SLA s; SRA s; SRL s; RLD; RRD	P/V=1 pentru paritate pară, altfel P/V=0
Test de bit	BIT b,s	Necunoscut
Intrare/ieșire	IN r,(C)	P/V=1 pentru paritate pară, altfel P/V=0
	INI; INIR; IND; INDR; OUTI; OTIR; OUTD; OTDR	Necunoscut

## 5. H, bitul indicator de transport la jumătate și N, bitul indicator de scădere

Indicatorii H și N nu pot fi testați de instrucțiuni de salt condiționat. Sînt utilizați de unitatea centrală pentru operații aritmetice BCD. Bitul H reprezintă transportul la jumătate, de la cei 4 biți mai puțin semnificativi ai rezultatului (cifra BCD mai puțin semnificativă) iar N este indicatorul de scădere, care arată dacă o adunare sau o scădere a fost tocmai executată. În general, o adunare sau o incrementare șterg bitul N, iar o scădere sau o decrementare îl înscriu (cu 1 logic). Cînd se execută instrucțiunea DAA după o adunare sau o scădere, se ține cont de starea bistabilelor N și H pentru a ajusta corect rezultatul binar la forma BCD. Pentru o adunare ( $N=0$ ), un rezultat binar trebuie corectat adunînd 6 la cifra BCD în anumite situații: dacă a existat un transport de la cifra BCD ( $H=1$  sau



C=1); dacă nu a existat transport, dar pe poziția cifrei BCD se află o valoare mai mare decât 1001<sub>B</sub>. Cîteva exemple sînt prezentate în continuare:)

19+	0001	1001	(19 BCD)
29	0010	1001	(29 BCD)
48	0100	0010	(42 BCD, eronat) C=0, H=1
	0000	0110	ajustează rezultatul, adunînd 6 la cifra BCD inferioară.
	0100	1000	(48 BCD, corect)
99+	1001	1001	(99 BCD)
99	1001	1001	(99 BCD)
198	1	0011	(32 BCD, eronat) C=1, H=1
		0110	ajustează rezultatul, adunînd 6 la ambele cifre BCD
	1	1001	(98 BCD, corect cu C=1)

În cazul unor scăderi (N=1) rezultatul binar trebuie corectat scăzînd 6 din cifra BCD, în anumite condiții: dacă H=1, se scade 6 din cifra BCD mai puțin semnificativă; dacă C=1, se scade 6 din cifra BCD mai semnificativă; dacă H=1 și C=1, se scade 6 din ambele cifre BCD. Două exemple vor fi date în continuare:

19-	0001	1001	(19 BCD)
91	1001	0001	(91 BCD)
-72	1	1000	(88 BCD, eronat) C=1, H=0
		0110	ajustează rezultatul, scăzînd 6 din cifra BCD superioară.
	1	0010	(28 BCD, corect, cu C=1)
11-	0001	0001	(11 BCD)
-99	1001	1001	(99 BCD)
-88	1	0111	(78 BCD, eronat) C=1, H=1
		0110	ajustează rezultatul, scăzînd 6 din ambele cifre BCD
	1	0001	(12 BCD, corect, cu C=1)

Aplicarea instrucțiunii DAA permite realizarea de operații aritmetice BCD în precizie multiplă, păstrînd transportul de la ultima adunare sau scădere BCD.

Instrucțiunile care afectează indicatorii de condiții H și N sînt cele din tabelul 3.6.

Acțiunile supra bitului indicator de transport la jumătate (H) și asupra bitului indicator de scădere (N)

Tabelul 3.6

Grup	Instrucțiuni	Acțiunea
Încărcare de 8 biți	LD A,I; LD A,R	N=0, H=0
Transfer de bloc și cântare	LDI; LDIR; LDD; LDDR	N=0, H=0
	CPI; CPID; CPD; CPDR	N=1; H=1 dacă există împrumut de la bitul 4, altfel H=0
Aritmetic de 8 biți	ADD A,s; ADC A,s	N=0; H=1 dacă există transport de la bitul 3, altfel H=0
	SUB s; SBC A,s	N=1; H=1 dacă există împrumut de la bitul 4, altfel H=0
	INC s	N=0; H=1 la transport din bitul 3, altfel H=0
	DEC s	N=1; H=1 dacă există împrumut de la bitul 4, altfel H=0



Grup	Instrucțiuni	Acțiunea
Logic de 8 biți	AND s; OR s; XOR s	N=0; H=1
	CP s	N=1; H=1 dacă există împrumut de la bitul 4, altfel H=0
Aritmetic general	DAA	N neafectat; H necunoscut.
	CPL	N=1; H=1
	NEG	N=1; H=1 dacă există împrumut de la bitul 4, altfel H=0
	CCF	N=0; H neafectat
	ISCF	N=1; H=0
Aritmetic de 16 biți	ADD HL,ss; ADC HL,ss	N=0; H=1 la transport din bitul 11, altfel H=0
	SBC HL,ss	N=1; H=1 dacă există împrumut de la bitul 12, altfel H=0
	ADD IX,pp; ADD IY,rr	N=0, H=1 dacă există transport de la bitul 11, altfel H=0
Rotire și deplasare	RLCA; RLA; RRCA; RRA; RLC s; RL s; RRC s; RR s; SLA s; SRA s; SRL s; RLD; RRD	N=0; H=0
Test de bit	BIT b,s	N=0; H=1
Intrare/ieșire	IN r,(C)	N=0; H=0
	INI; INIR; IND; INDR; OUTI; OTIR; OUTD; OTDR	N=1; H necunoscut

### 3.3. EXEMPLE DE UTILIZARE A INSTRUCȚIUNILOR SPECIFICE MICROPROCESORULUI Z80

#### 1. Transferul unui bloc de date

Presupunând că un șir de date din memorie, începând la locația cu adresa "DATE" trebuie transferat în zona de memorie începând cu adresa "TAMPON" și blocul de date cu lungimea de 737 octeți, următoarele instrucțiuni realizează operația dorită, programul având lungimea de 11 octeți și durata de 21 perioade de tact pentru fiecare octet transferat.

```
LD    HL,DATE      ; INCARCA ADRESA DE INCEPUT A BLOCULUI
                     ; DE DATE IN HL
LD    DE,TAMPON    ; INCARCA ADRESA DE INCEPUT A ZONEI DE
                     ; DESTINATIE IN DE
LD    BC,737       ; INCARCA LUNGIMEA BLOCULUI DE DATE IN
                     ; BC
LDIR                     ; TRANSFERA BLOCUL DE DATE (DE LA ADRESA
                     ; HL LA DE, INCREMENTIND HL SI DE SI DE-
                     ; CREMENTIND BC, PINA CIND BC=0)
```

#### 2. Transferul condiționat al unui bloc de date

Să presupunem că trebuie efectuat transferul unui bloc de date din memorie, începând de la adresa "DATE", în zona de memorie care începe cu adresa "TAMPON" transferul având loc pînă cînd se găsește în șirul de date codul ASCII al caracterului "\$", utilizat ca delimitator al șirului de date. Se mai presupune că lungimea ma-



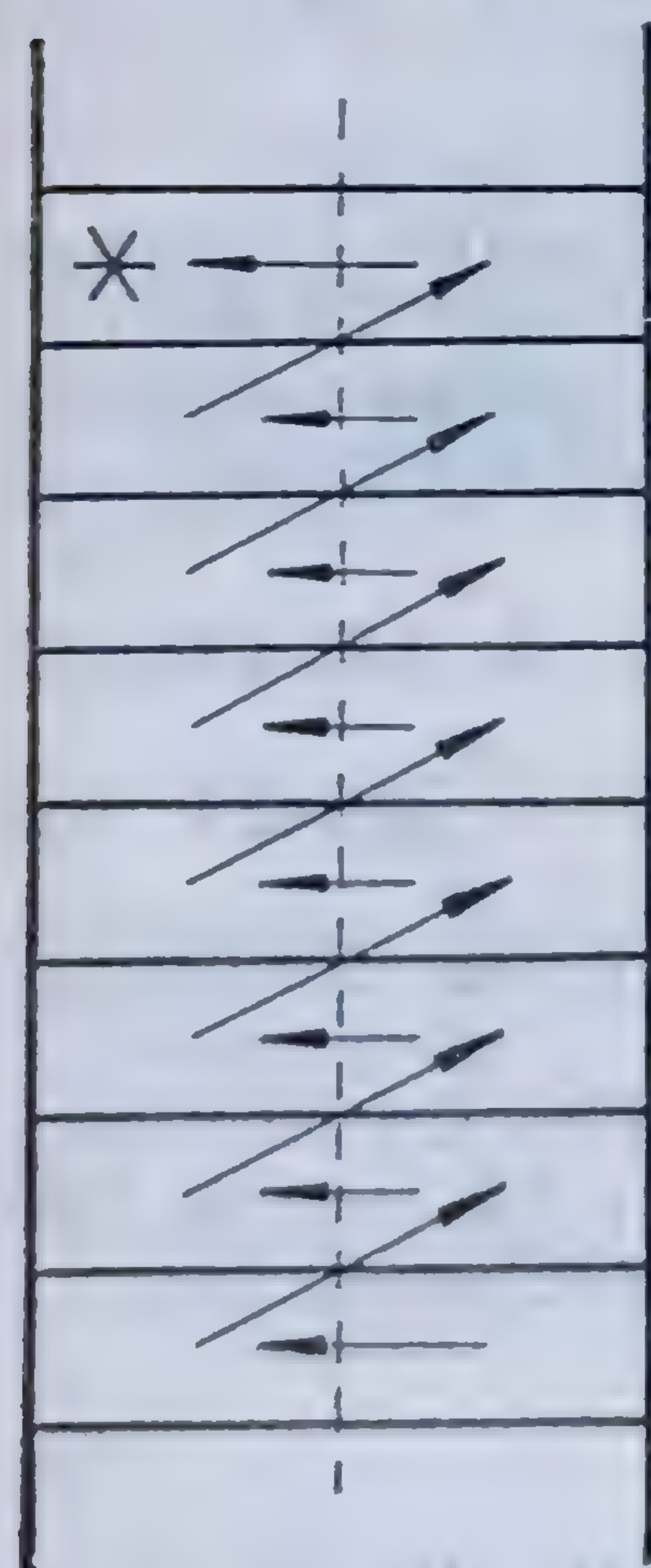
ximă a unui bloc de date este de 132 caractere. Programul care realizează transferul, avînd o lungime de 19 octeți, este următorul:

LD HL,DATE	; INCARCA ADRESA DE INCEPUT A BLOCULUI DE DATE IN HL
LD DE,TAMPON	; INCARCA ADRESA DE INCEPUT A ZONEI DE DESTINATIE IN DE
LD BC,132	; INCARCA LUNGIMEA MAXIMA DE BLOC IN BC
BUCLA: LD A,"\$"	; INCARCA IN A CODUL ASCII AL DELIMITATORULUI DE BLOC
CP (HL)	; COMPARA CONTINUTUL MEMORIEI CU CODUL DELIMITATORULUI
JR Z,END1	; SALT LA ADRESA END 1 (SFIRSIT) CIND CODUL DE MEMORIE ESTE "\$"
LDI	; TRANSFERA CHARACTERUL DE LA ADRESA HL LA ADRESA DE; HL+1 →
	; →HL,DE+1 → DE,BC-1 → BC (DACA P/V=1)
END: JP PE,BUCLA	; SALT LA ADRESA 'BUCLA' DACA MAI EXISTA CARACTERE, ALTFEL CONTINUA
END1: .....	; PRELUCREAZA BLOCUL DE DATE TRANSFERAT SAU EFECTUEAZA ALTE OPERATII

De notat că bitul P/V este utilizat pentru a indica dacă BC a fost decrementat pînă la zero.

### 3. Deplasarea unui șir de cifre BCD

În exemplul următor, se realizează deplasarea unui șir de cifre BCD (2 cifre/octet), ca în fig. 3.11. Operația este necesară în realizarea subrutinelor de multiplicare sau divizare BCD, și este efectuată de programul următor, cu o lungime de 11 octeți.



LD HL,DATE	; ADRESA PRIMULUI OCTET IN HL
LD B,NUMAR	; NUMARUL DE OCTETI DEPLASATI, IN B
XOR A	; STERGE ACUMULATORUL
ROT: RLD	; ROTESTE LA STINGA CIFRELE BCD DIN A3-A0 (HL)
INC HL	; INCREMENTEAZA INDICATORUL DE MEMORIE
DJNZ ROT	; DECREMENTEAZA B.SALT LA ADRESA "ROT" DACA B ≠ 0, ALTFEL CONTINUA

Fig. 3.11 Deplasarea cifrelor BCD în locații adiacente de memorie.

După ultima rotire, jumătatea inferioară a acumulatorului conține cifra BCD marcată cu "\*" în figura 3.11. Pentru aceeași figură, B = 8 în programul de mai sus.



#### 4. Scăderea a două numere BCD

Exemplul următor realizează scăderea a două numere BCD, de lungime egală dar variabilă, rezultatul fiind plasat în locul descăzutului. Programul, cu o lungime de 17 octeți, este următorul:

```
LD HL,ADRS(0) ; INCARCA ADRESA SCAZATORULUI IN HL
; (INDICA CIFRELE CELE MAI PUTIN
; SEMNIFICATIVE)

LD DE,ADRD ; INCARCA ADRESA DESCAZUTULUI IN DE
; (INDICA CIFRELE CELE MAI PUTIN
; SEMNIFICATIVE)

LD B,L ; INCARCA IN B LUNGIMEA OPERANZILOR,
; IN NUMAR DE OCTETI (2 CIFRE BCD/
; OCTET)

AND A ; STERGE BITUL DE TRANSPORT, C
SCZEC: LD A,(DE) ; 2 CIFRE BCD DIN DESCAZUT, IN
; ACUMULATOR
SBC A,(HL) ; SCADE (HL) DIN ACUMULATOR
DAA ; AJUSTEAZA ZECIMAL ACUMULATORUL
LD (HL),A ; DEPUNE CELE 2 CIFRE BCE ALE REZUL-
; TATULUI IN LOCUL CELOR DIN SCAZATOR
INC HL ; INCREMENTEAZA HL PENTRU A INDICA
; URMATOARELE 2 CIFRE BCD DIN
; SCAZATOR
INC DE ; IDEM, PENTRU DESCAZUT (SE INDICA 2
; CIFRE MAI SEMNIFICATIVE)
DJNZ SCZEC ; DECREMENTEAZA B SI EFECTUEAZA SALT
; LA "SCZEC" DACA B≠0, ALTFEL CONTINUA
```

#### 5. Ordonarea unui șir de numere

În continuare este prezentat un program care ordonează un șir de numere, fiecare în intervalul [0, 255]. La intrarea în program, HL conține adresa de început a șirului de date și C conține numărul de elemente care trebuie adunate, iar la ieșire se regăsește șirul de numere în aceleași locații, dar ordonat descrescător. În timpul execuției programului, A conține temporar date sau rezultate ale calculelor, B este numărător pentru zona de date, C conține lungimea șirului de date, D conține primul element care se compară, E conține al doilea element care se compară, H conține un indicator pentru executarea unui schimb, IX este un indicator pentru zona de date, iar L și IY sînt neutilizate. Programul, scris ca subrutină, se prezintă asamblat de la adresa 0000H și este următorul:

ADRESA	CODUL ETICHETA	COD	OPERATIE	COMENTARIU
0000	222600	SORT:	LD (DATA),HL	SALVEAZA ADRESA DE IN- CEPUT A DATELOR LA 0026H
0003	CB84	BUCLA:	RES 0,H	INITIALIZEAZA LA 0 BITUL 0 DIN H, INDICATOR DE SCHIMB
0005	41		LD B,C	INITIALIZEAZA NUMARA- TORUL DE DATE
0006	05		DEC B	DECREMENTEAZA B PEN- TRU TESTARE
0007	DD2A2600		LD IX, (DATA)	ADRESA DE INCEPUT A DATELOR, IN IX



000B	DD7E00	URM:	LD A,(IX+0)	; PRIMUL ELEMENT DE COM-
				; PARAT, ADUS IN A
000E	57		LD D,A	; PRIMUL ELEMENT, TRANS-
				; FERAT TEMPORAR IN D
000F	DD5E01		LD E,(IX+1)	; AL DOILEA ELEMENT DE
				; COMPARAT, ADUS IN E
0012	93		SUB E	; COMPARA PRIMUL CU AL
				; DOILEA ELEMENT, PRIN
				; SCADERE; C=1 DACA PRI-
				; MUL < AL DOILEA
0013	3008		JR NC,NUSCH	; DACA PRIMUL < AL DOILEA
				; CONTINUA SECVENTIAL
0015	DD7300		LD (IX+0),E	; SCHIMBA INTRE ELE ELE-
				; MENTELE COMPARATE
0018	DD7201		LD (IX+1),D	
001B	CBC4		SET 0,H	; INREGISTREAZA SCHIMBA-
				; REA FACUTA, PRIN
				; BIT0=1 IN H
001D	DD23	NUSCH:	INC IX	; INDICA URMATORUL ELE-
				; MENT DE COMPARAT
001F	10EA		DJNZ URM	; DECREMENTEAZA B PT, A
				; NUMARA COMPARARILE
				; FACUTE; SALT LA "URM"
				; DACA MAI EXISTA PERECHI
				; DE DATE
0021	CB44		BIT 0,H	; TESTEAZA BITUL 0 DIN H
				; PENTRU A DETERMINA DA-
				; CA S-A EFECTUAT UN
				; SCHIMB (H0→Z)
0023	20DE		JR NZ,BUCLA	; CONTINUA DACA DATELE
				; NU SINT ORDONATE, CU
				; SALT LA "BUCLA"
0025	C9		RET	; REVINE
0026	XXXX			; 2 LOCATII PT. ADRESA DE
				; INCEPUT A DATELOR

Programul compară, începînd cu primele 2 elemente, perechile de elemente alăturate, și eventual schimbă ordinea celor 2 elemente comparate dacă nu sînt în ordine descrescătoare. Pentru N elemente în șir, se fac N — 1 comparări, numărate în registrul B. N — 1 comparări se fac prin parcurgerea de N — 1 ori a buclei care începe la adresa URM și se termină cu DJNZ URM. Dacă la un set de comparări s-au schimbat între ele cel puțin 2 elemente (H0=1), se reiau comparările începînd cu adresa "BUCLA". Dacă în urma a N—1 comparări, H0=0, șirul este în ordine/descrescătoare și se revine în programul principal.

## 6. Multiplicarea a 2 numere întregi de 16 biți

Subrutina de mai jos primește la intrare deînmulțitul în HL și înmulțitorul în DE. La ieșire, rezultatul se află în HL. În timpul execuției programului, H și L conțin rezultatul parțial, DE deînmulțitul, B, numărul de deplasări, C și A înmulțitorul (octet superior, respectiv inferior). Condiția ca rezultatul să fie corect este să nu depășească valoarea 65535 (capacitatea a 2 octeți). Subrutina, cu o lungime de 20 octeți, este prezentată în continuare.



ADRESA	CODUL	ETICHETA	COD OPERATIE	COMENTARIU
0000	0610	INM:	LD B,16	; NUMARUL DE BITI IN B
0002	4A		LD C,D	; OCTET SUPERIOR INMULTI-
0003	7B		LD A,E	; TOR IN C
0004	EB		EX DE,HL	; OCTET INFERIOR INMULTI-
0005	210000		LD HL,0	; TOR IN A
0008	CB39	BUCLA:	SRL C	; DEINMULTITUL IN DE SI IN-
000A	1F		RR A	; MULTITORUL IN HL
000B	3001		JR NC,NUAD	; STERGE REZULTATUL PARTI-
000D	19		ADD HL,DE	; AL
000E	EB	NUAD:	EX DE,HL	; DEPLASEAZA INMULTITORUL
000F	29		ADD HL,HL	; SPRE DREAPTA
0010	EB		EX DE,HL	; CEL MAI PUTIN SEMNIFICA-
0011	10F5		DJNZ BUCLA	; TIV BIT ESTE IN BITUL C,
0013	C9		RET	; DE TRANSPORT
				; DACA C=0, NU EFECTUEAZA
				; ADUNAREA
				; DACA C=1, ADUNA DEINMUL-
				; TITULA REZULTATUL
				; PARTIAL
				; DEPLASEAZA DEINMULTITUL
				; SPRE STINGA, PRIN INMUL-
				; TIRE CU 2
				; REPETA PINA LA TERMINA-
				; REA NUMARULUI DE BITI
				; (DECREMENTEAZA SI TESTEA-
				; ZA B)
				; REVINE IN PROGRAMUL
				; PRINCIPAL



## CAPITOLUL IV

### CIRCUITUL NUMĂRĂTOR-TEMPORIZATOR Z80 CTC

#### 4.1. DESCRIEREA CIRCUITULUI NUMĂRĂTOR-TEMPORIZATOR Z80 CTC

Circuitul conține 4 canale care pot fi programate pentru aplicații de numărare-temporizare, independent unul de altul. Se pot număra evenimente, temporiza întreruperi sau genera diferite frecvențe. Circuitul se poate conecta direct atât la unitatea centrală, cât și la circuitul de interfață serie.

Circuitul CTC conține 4 elemente importante: interfața de intrare/ieșire pentru magistrala de date a unității centrale, logică de control a canalelor, logică de întreruperi și circuite numărătoare-temporizatoare (fig. 4.1.).

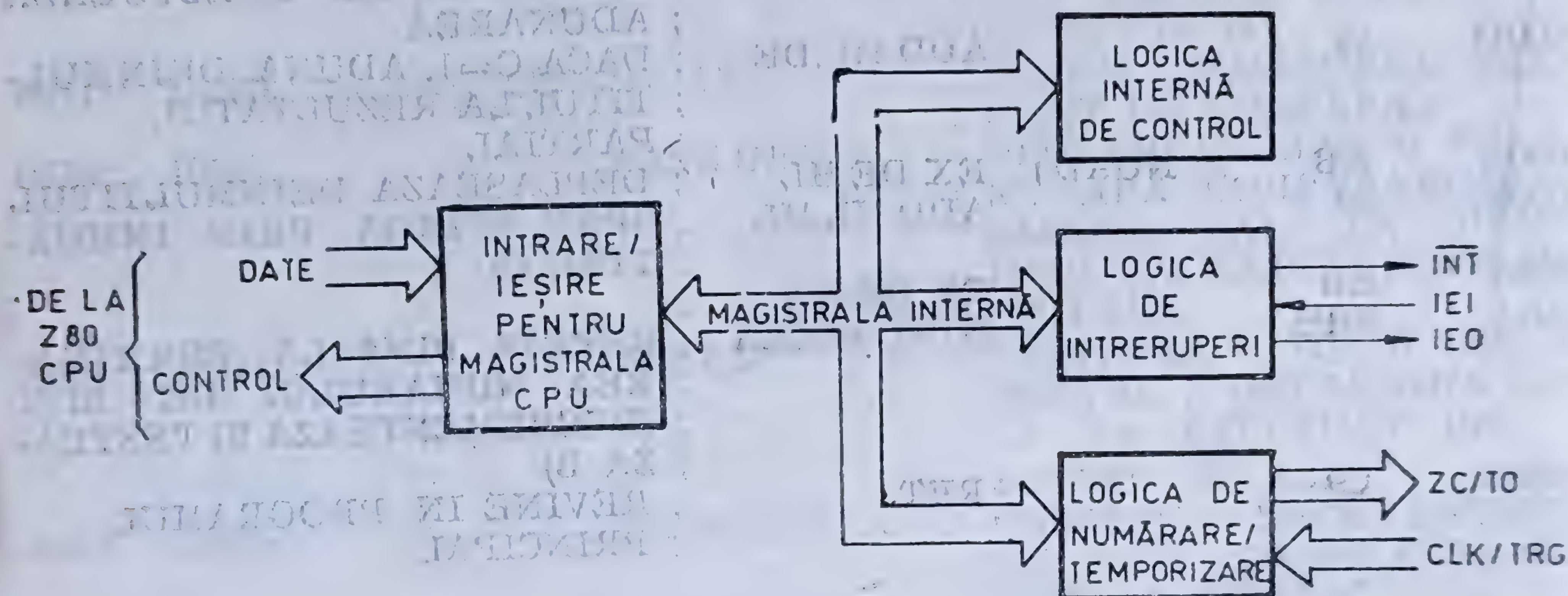


Fig. 4.1 Diagrama bloc funcțională a circuitului Z80 CTC.

Circuitul de I/E pentru magistrala de date decodifică intrările de adrese și interfațează datele și semnalele de control ale CPU, pentru distribuire pe magistrala internă.

Logica de control se referă la activarea circuitului, la inițializarea lui și la operațiile de scriere/citire.

Logica de întreruperi controlează prioritatea la întreruperi a circuitului, ca o funcție de semnalul IEI. Dacă  $IEI=1$ , circuitul CTC poate cere o întrerupere. În timpul prelucrării întreruperii, IEO este menținută la 0 logic, inhibând cererile de întrerupere ale dispozitivelor mai puțin prioritare. Dacă  $IEI=0$ , circuitul nu poate solicita o întrerupere și fixează și  $IEO=0$ .

Dacă un canal este programat să ceară o întrerupere, logica de întreruperi face  $IEO=0$ , la atingerea numărului 0 în timpul numărării, și generează un semnal  $\overline{INT}$  spre unitatea centrală. Când aceasta răspunde cu recunoașterea întreruperii ( $\overline{M1}$  și  $\overline{IORQ}$ ), logica de întreruperi arbitrează prioritățile interne și plasează un vector unic de întreruperi pe magistrala de date.

Dacă o întrerupere așteaptă servirea, logica de întreruperi menține  $IEO=0$ . Când unitatea centrală emite o instrucțiune de revenire din întrerupere (RETI), fiecare dispozitiv periferic decodifică primul octet ( $ED_n$ ). Dacă dispozitivul așteaptă



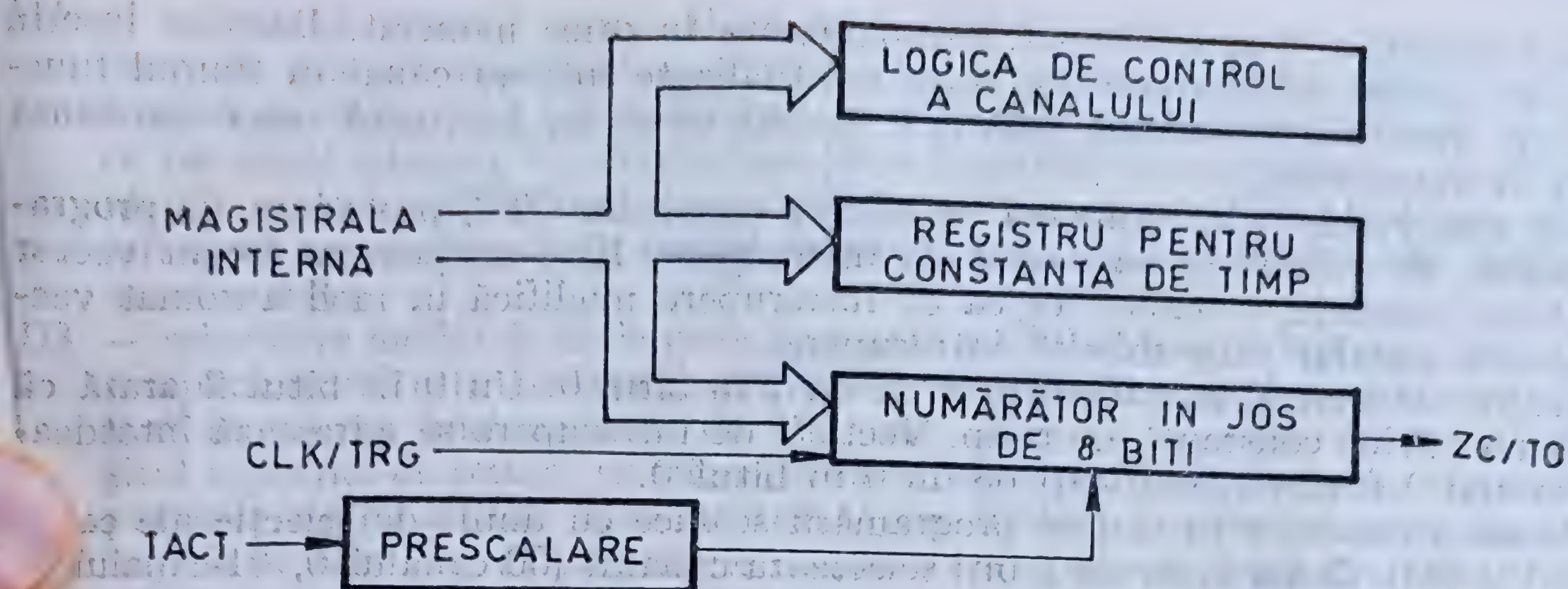


Fig. 4.2 Diagrama bloc a unui numărător/temporizator.

servirea (acceptarea) unei întreruperi, face  $IEO=1$  timp de un ciclu M1. Aceasta face posibil ca toate dispozitivele cu prioritate mai mică să poată decodifica întreaga instrucțiune RETI și să se inițializeze în mod corect.

Circuitele numărătoare-temporizatoare au structura reprezentată în fig. 4.2.

Logica de control a canalului primește cuvântul de control de 8 biți când se programează canalul, decodifică cuvântul de control și fixează condițiile de validare sau nu a întreruperilor, modul de funcționare (numărător sau temporizator) factorul de prescalare în modul de temporizator (16 sau 256), frontul activ pentru intrarea CLK/TRG, declanșarea în modul de temporizator (automat, sau cu intrarea CLK/TRG), apariția unui cuvânt de date pentru constanta de timp, sau inițializarea prin program.

Registrul constantei de timp primește și depozitează constanta de timp de 8 biți la programarea canalului. Valoarea ei este în domeniul  $1 \div 256$  și este automat încărcată în numărător când este inițializat canalul și în continuare, după fiecare atingere a valorii 0.

Prescalarea se efectuează doar în modul de temporizator și constă în divizarea frecvenței de tact a sistemului cu 16 sau 256. Ieșirea circuitului de prescalare constituie intrare pentru numărătorul descrescător în timpul funcționării ca temporizator. Efectul este multiplicarea perioadei de tact a sistemului cu 16 sau 256.

Numărătorul descrescător al canalului este încărcat cu constanta din registrul constantei de timp, înaintea fiecărui ciclu de numărare. Numărătorul este decrementat într-unul dintre cele două moduri posibile, după modul de funcționare: de ieșirea divizorului de prescalare (modul de temporizator) sau de impulsurile aplicate pe intrarea CLK/TRG (modul de numărător). Unitatea centrală poate citi conținutul numărătorului în curs de decrementare în orice moment, fără a afecta numărarea, printr-o operație de citire de la adresa portului asociat canalului din circuitul CTC. Când numărătorul atinge valoarea 0, ieșirea ZC/TO generează un impuls pozitiv. Când întreruperile sînt validate, atingerea lui 0 generează de asemenea și o cerere de întrerupere ( $\overline{INT}$ ). În modul de temporizator, intervalul minim între două impulsuri la ieșire este de  $6,4 \mu s$  (Z80) sau  $4 \mu s$  (Z80A).

### Programarea circuitului CTC

Fiecare canal trebuie programat înainte de funcționare. Programarea constă în scrierea a două cuvinte în portul de I/E care corespunde canalului. Primul este un cuvânt de control care selectează modul de funcționare și alți parametri. Al doilea cuvânt este o constantă de timp de 8 biți, cu valoarea între 1 și 256. Un cuvânt constantă de timp trebuie să fie precedat de un cuvânt de control al canalului.



După inițializare, canalele pot fi programate în orice moment. Dacă se înscriu cuvinte de control și constante de timp reactualizate într-un canal în timpul funcționării lui, numărarea continuă până la 0 înainte ca să fie încărcată noua constantă de timp în numărător.

Dacă este validată întreruperea pe oricare canal din CTC, procedura de programare trebuie să includă și un vector de întrerupere. Este necesar un singur vector pentru toate canalele, deoarece logica de întrerupere modifică în mod automat vectorul pentru canalul care solicită întreruperea.

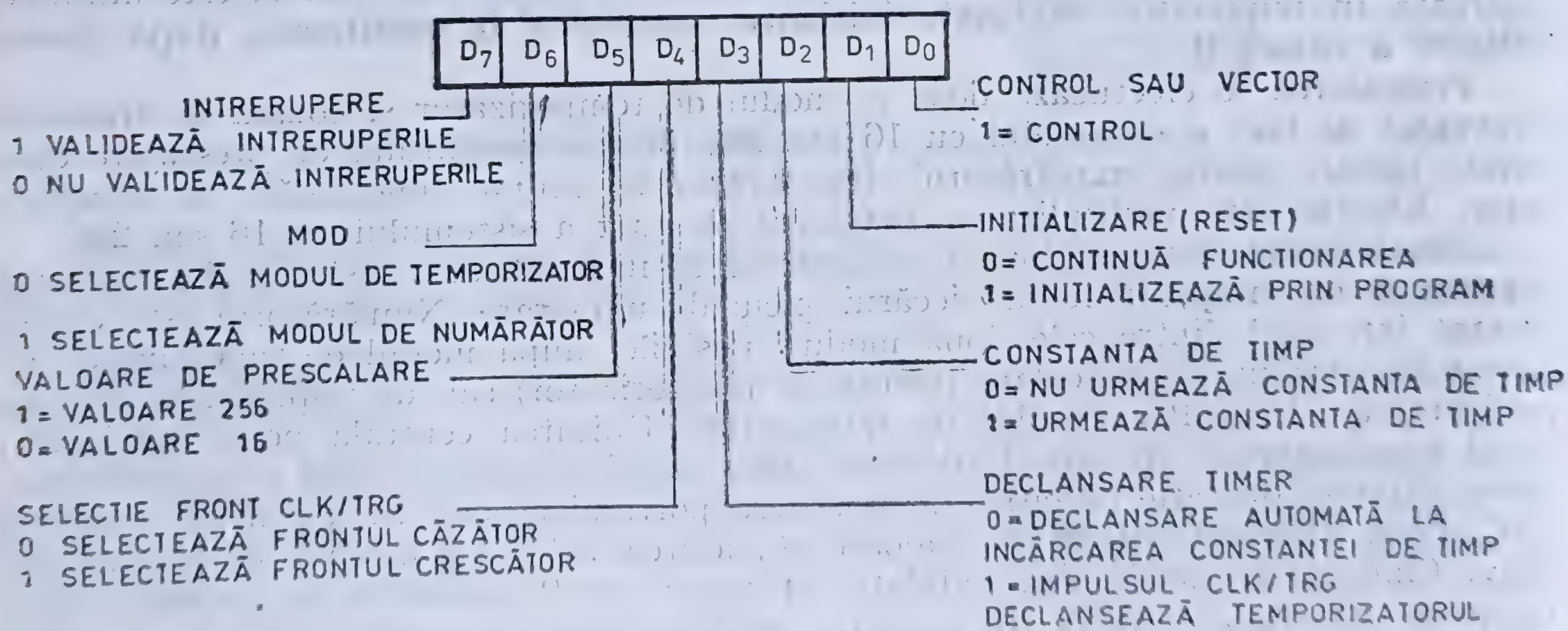
Cuvîntul de control este identificat de un 1 în bitul 0. Un 0 în bitul 2 arată că urmează un cuvînt constantă de timp. Vectorii de întrerupere se adresează întotdeauna canalului 0 și sînt identificați de un 0 în bitul 0.

Adresarea canalelor în timpul programării se face cu liniile de selecție ale canalelor, CS1 și CS0. Codul binar de 2 biți selectează canalul (00-canalul 0, 01-canalul 1, 10-canalul 2, 11-canalul 3).

Inițializarea circuitului poate fi realizată prin semnalul de inițializare sau prin program. În prima variantă se termină toate numărările în jos și se dezactivează toate întreruperile, ștergînd biții de întrerupere în registrele de control. În plus, ieșirile ZC/TO și INT devin inactive, IEO=IEI și D0-D7 trec în starea de impedanță ridicată. Toate canalele trebuie reprogramate complet după această inițializare.

Inițializarea prin program este controlată de bitul 1 din cuvîntul de control al canalului. Cînd un canal primește un semnal de inițializare prin program, se oprește din numărare. După inițializarea prin program, un nou cuvînt constantă de timp trebuie înscris în același canal. Cuvîntul care realizează inițializarea prin program afectează și ceilalți biți ai registrului de control al canalului.

Dacă biții D1 și D2 ai cuvîntului de control al canalului sînt 1, canalul adresat se oprește din funcționare, așteptînd un nou cuvînt constantă de timp. Canalul este gata de reluarea funcționării după ce noua constantă a fost programată. În modul de temporizator, dacă D3 = 0, funcționarea este începută automat, la încărcarea cuvîntului constantă de timp.



Programarea cuvîntului de control al canalului (fig. 4.3) se face după cum urmează:

- D7 — validează întreruperile, ca o cerere de întrerupere (INT) să fie generată cînd numărătorul atinge valoarea 0. Întreruperile pot fi programate în oricare dintre moduri și pot fi validate sau nu în orice moment.
- D6 — selectează modul de funcționare de temporizator sau de numărător.
- D5 — selectează factorul de prescalare 16 sau 256, numai în modul de temporizator



- D4 — selectează frontul activ al impulsurilor CLK/TRG; reprogramarea acestui front în timpul funcționării echivalează cu injectarea unui front activ; dacă frontul de declanșare este schimbat de un cuvânt de control reactualizat în timp ce un canal așteaptă intrarea în funcțiune în modul de temporizator, rezultatul este identic cu cel realizat de un impuls CLK/TRG și temporizatorul este pornit; similar, dacă un canal este în modul de numărător, conținutul lui este decrementat cu o unitate.
- D3 — selectează modul de declanșare (numai în modul de temporizator); când  $D3 = 0$ , temporizatorul este declanșat automat; cuvântul constantă de timp este programat în timpul unei operații de I/E, care durează un ciclu de mașină; la sfârșitul operației de scriere există o întârziere de o perioadă de tact; temporizatorul pornește automat (se decrementează) pe frontul pozitiv al celui de al doilea impuls de tact ( $T_2$ ) al ciclului de mașină care urmează după operația de scriere; odată pornit, temporizatorul funcționează continuu; la atingerea lui 0, temporizatorul reîncarcă automat constanta și continuă numărarea fără întrerupere sau întârziere, până la oprirea printr-o inițializare; când  $D3=1$ , temporizatorul este declanșat extern, prin intrarea CLK/TRG; cuvântul constantă de timp este programat în timpul unei operații de I/E, care durează un ciclu de mașină; temporizatorul este gata de funcționare pe frontul pozitiv al celui de al doilea impuls de tact ( $T_2$ ) al următorului ciclu de mașină; pentru declanșare imediată, semnalul de pe intrarea CLK/TRG trebuie să preceadă  $T_2$  cu o perioadă de tact plus un timp minim de prestabilire; dacă acest minim nu este îndeplinit, temporizatorul va porni în a treia perioadă de tact ( $T_3$ ); odată pornit, funcționează continuu, până la oprirea printr-o inițializare.
- D2 — arată dacă urmează o constantă de timp; dacă  $D2 = 1$ , următorul cuvânt adresat canalului selectat este o constantă de timp, destinat registrului constantei de timp; cuvântul constantă de timp poate fi înscris în orice moment; dacă  $D2 = 0$ , nu urmează o constantă de timp; această situație apare de obicei când canalul funcționează deja și noul cuvânt de control al canalului este o reactualizare; nici un canal nu va funcționa fără o valoare a constantei de timp; singurul mod de a înscrie o constantă de timp este de a înscrie anterior un cuvânt de control cu  $D2 = 1$ .
- D1 — determină o inițializare prin program, dacă  $D1 = 1$ .
- D0 — identifică cuvântul de control, dacă  $D0 = 1$ .

Programarea constantei de timp trebuie efectuată pentru ca să înceapă numărarea unui canal. În timpul programării sau reprogramării, un cuvânt de control al canalului cu  $D2 = 1$  precede cuvântul constantă de timp, pentru a arăta că următorul cuvânt este o constantă de timp. Constanta de timp poate avea orice valoare de la 1 la 256 (fig. 4.4). Valoarea  $00_H$  este interpretată ca  $256_H$ .

În modul de temporizator, intervalul de timp este controlat de trei factori: perioada de tact a sistemului, factorul de prescalare (P), care multiplică perioada cu 16 sau 256, și constanta de timp (T), care este programată în registrul constantei de timp. Ca urmare, intervalul de timp este produsul  $\Phi \times P \times T$  ( $\Phi$  este perioada de tact). Rezoluția minimă a temporizatorului este  $16 \times \Phi$  ( $4 \mu s$  cu un semnal de tact de

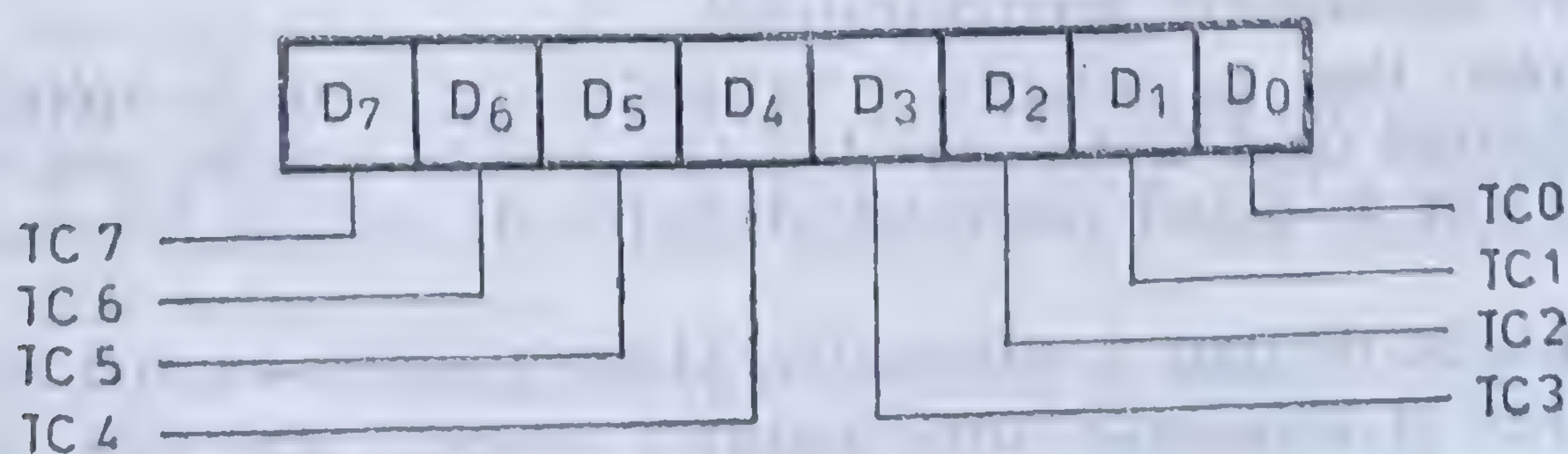


Fig. 4.4 Cuvântul constantă de timp.



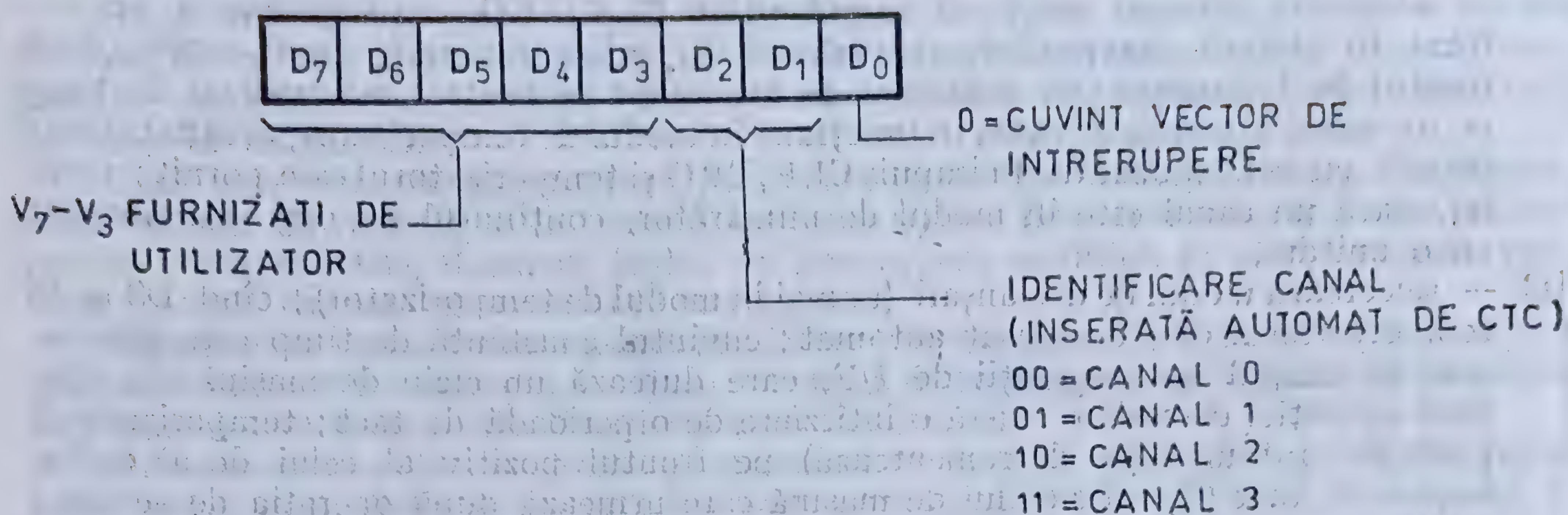


Fig. 4.5 Cuvînt vector de întrerupere.

4 MHz). Intervalul maxim al temporizatorului este  $256 \times \Phi \times 256$  (16,4 ms cu un tact de 4 MHz). Pentru intervale mai lungi, temporizatoarele pot fi legate în cascadă.

Programarea vectorului de întreruperi trebuie efectuată dacă circuitul Z80 CTC are validate una sau mai multe întreruperi. Pentru a putea furniza vectorul de întreruperi spre unitatea centrală, circuitul CTC trebuie să fie programat cu cei 5 biți mai semnificativi ai vectorului de întreruperi. Programarea constă din înscrierea unui cuvînt vector în portul de I/E corespunzînd canalului 0 al circuitului CTC. Bitul D0 al cuvîntului vector este întotdeauna 0, pentru a distinge vectorul de un cuvînt de control al canalului. Biții D1 și D2 nu sînt utilizați în programarea cuvîntului vector. Ei sînt furnizați de logica de întreruperi pentru identificarea canalului care a cerut servirea prin întrerupere, cu un vector de întrerupere unic (fig. 4.5). Canalul 0 are cea mai mare prioritate.

### Conexiunile circuitului Z80 CTC

Funcțiile logice ale circuitului sînt reprezentate în figura 4.6. Descrierea lor este următoarea:

$\overline{CE}$  — intrare; cînd este activat prin  $\overline{CE} = 0$ , circuitul CTC acceptă cuvinte de control, vectori de întrerupere sau cuvinte constantă de timp, de pe magistrala de date, în timpul unui ciclu de scriere de I/E; în timpul unui ciclu de citire de I/E, se poate transmite unității centrale conținutul unui numărator descrescător; în cele mai multe aplicații, acest semnal de selecție a circuitului este decodificat din cei 8 biți inferiori ai magistralei de adrese.

CLK — intrare de tact, pe care se aplică semnalul de tact cu o singură fază al sistemului Z80.

CLK/TRG<sub>0</sub> — CLK/TRG<sub>3</sub> — intrări, active pe 0 sau pe 1, conform selecției utilizatorului; sînt semnale de tact extern/declanșare temporizator; cele 4 intrări corespund celor 4 canale din circuitul Z80 CTC; în modul de numărator, fiecare front activ pe acest pin decrementează număratorul; în modul de temporizator, un front activ declanșează temporizatorul.

CS0—CS1 — intrări, linii de selecție a canalului, pe care se aplică un cod binar de 2 biți, selectînd unul dintre cele 4 canale pentru o citire sau scriere de I/E; aceste intrări sînt de obicei controlate de liniile de adresă A0 și A1 ale unității centrale.

D0—D7 — magistrala de date a sistemului, bidirecțională, cu trei stări, pe care se transferă datele și comenzile între unitatea centrală Z80 CPU și circuitul Z80 CTC.



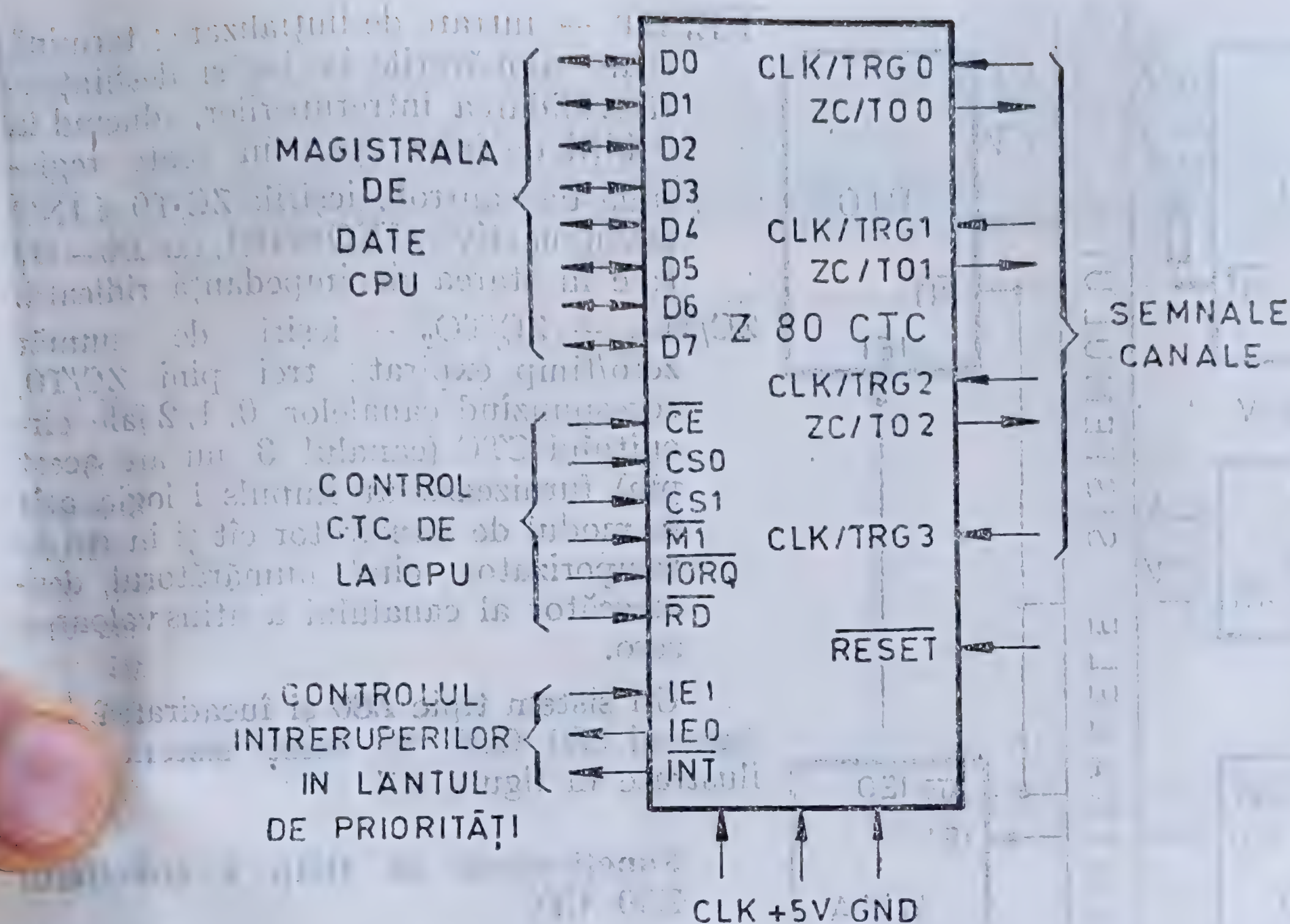


Fig. 4.6 Funcțiile logice ale circuitului Z80-CTC.

**IEI** — intrare de validare a întreruperilor;  $IEI = 1$  arată că nici un alt dispozitiv cu prioritate mai mare din lanțul de priorități nu este deservit de unitatea centrală în cadrul unei rutine de întrerupere.

**IE0** — ieșire de validare a întreruperilor;  $IE0 = 1$  numai dacă  $IEI = 1$  și unitatea centrală nu deservește nici o întrerupere de la un canal al circuitului Z80 CTC;  $IE0 = 0$  nu permite dispozitivelor cu prioritate mai mică să întrerupă unitatea centrală în timp ce un dispozitiv cu prioritate mai mare este servit în cadrul unei rutine de întrerupere.

**INT** — ieșire, cu drenă în gol, pentru cereri de întrerupere;  $INT = 0$  când oricare canal care a fost programat cu validarea întreruperilor a atins valoarea 0, în numărătorul descrescător al canalului.

**IORQ** — intrare pentru cerere de I/E; se utilizează în combinație cu  $\overline{CE}$  și  $\overline{RD}$  pentru a transfera date și cuvinte de control al canalelor între unitatea centrală Z80 CPU și circuitul Z80 CTC; în timpul unui ciclu de scriere,  $IORQ$  și  $\overline{CE}$  sînt active, iar  $\overline{RD}$  inactiv; circuitul Z80 CTC nu primește un semnal de scriere specific; acest semnal se generează intern ca inversul unui semnal activ  $\overline{RD}$ ; într-un ciclu de citire,  $IORQ$ ,  $\overline{CE}$  și  $\overline{RD}$  sînt active; conținutul numărătorului descrescător selectat prin liniile  $CS0$  și  $CS1$  este citit de unitatea centrală; dacă  $IORQ$  și  $\overline{M1}$  sînt ambele active, CPU anunță recunoașterea unei întreruperi și canalul cu cea mai mare prioritate care a cerut o întrerupere își plasează vectorul de întrerupere pe magistrala de date Z80.

**$\overline{M1}$**  — intrare, care indică primul ciclu de mașină; când  $\overline{M1}$  și  $IORQ$  sînt active, se anunță recunoașterea unei întreruperi; circuitul Z80 CTC plasează vectorul de întrerupere pe magistrala de date dacă are cea mai mare prioritate și dacă un canal a cerut o întrerupere.

**$\overline{RD}$**  — intrare de citire; este utilizată în combinație cu  $IORQ$  și  $\overline{CE}$  pentru a transfera date și cuvinte de control al canalului între unitatea centrală Z80 CPU și circuitul Z80 CTC; când nu este activă, permite operații de scriere.



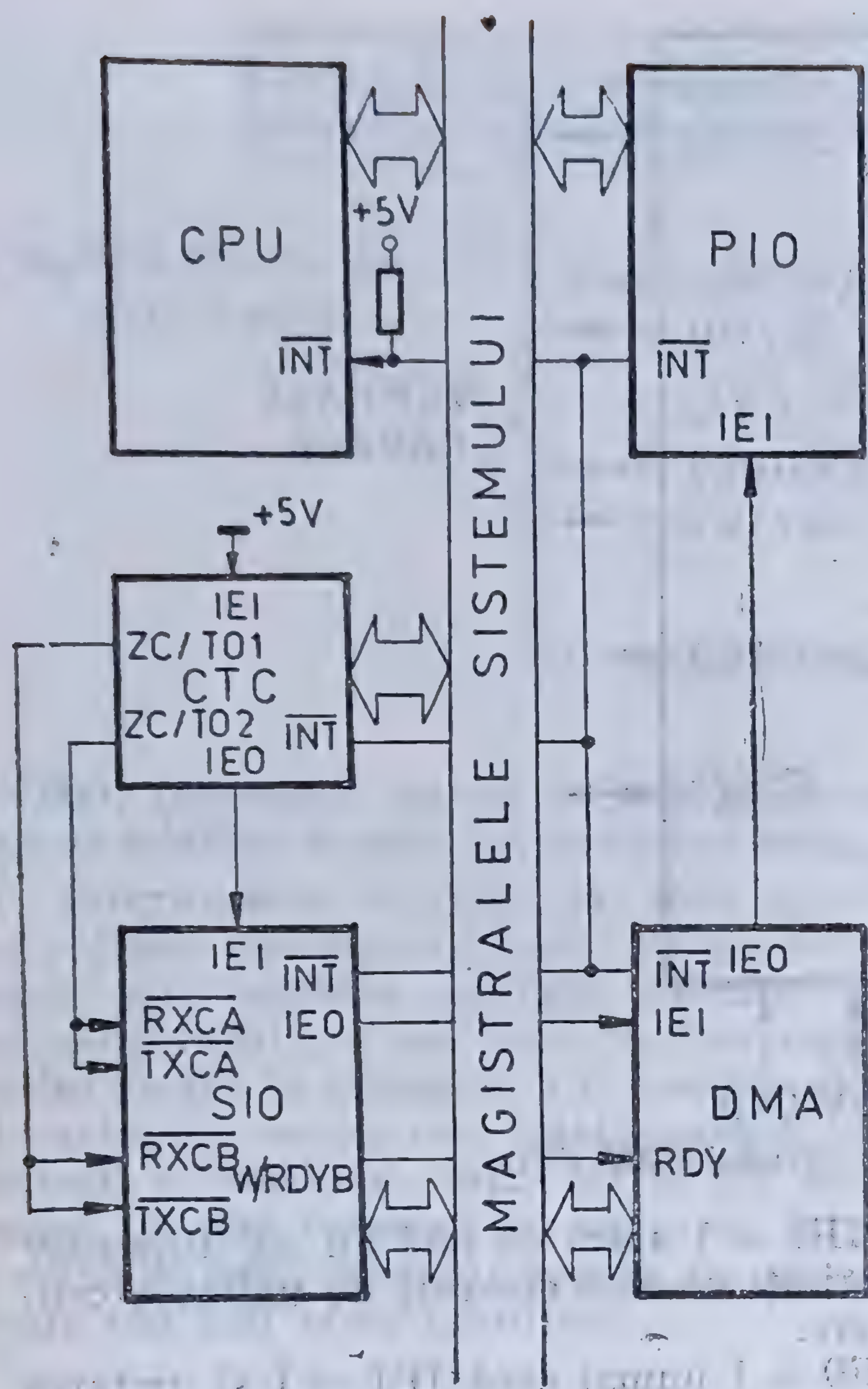


Fig. 4.7 Sistem Z80 tipic.

front crescător al lui CLK pentru a valida numărătorul de prescalare pe următorul front al tactului. Dacă frontul lui CLK/TRG apare mai aproape, funcționarea temporizatorului va fi întârziată cu o perioadă de tact. Temporizatorul poate fi pornit și automat dacă este astfel programat prin cuvântul de control al canalului.

În modul de numărător, impulsul CLK/TRG decrementează numărătorul descrescător. Semnalul de declanșare este asincron, dar numărarea este sincronizată cu semnalul CLK. Pentru ca decrementarea să apară pe următorul front crescător al lui CLK, frontul semnalului de declanșare trebuie să preceadă semnalul CLK cu un timp minim, ca în fig. 4.9 (LEAD TIME). Dacă acest timp este mai mic decât o valoare prescrisă, numărarea este întârziată cu o perioadă de tact. Impulsul de declanșare trebuie să aibă o lățime minimă și perioada impulsului de declanșare trebuie să fie cel puțin de două ori perioada impulsurilor de tact. Ieșirea ZC/TO apare imediat după ce numărătorul atinge valoarea 0, și urmează frontului crescător al lui CLK.

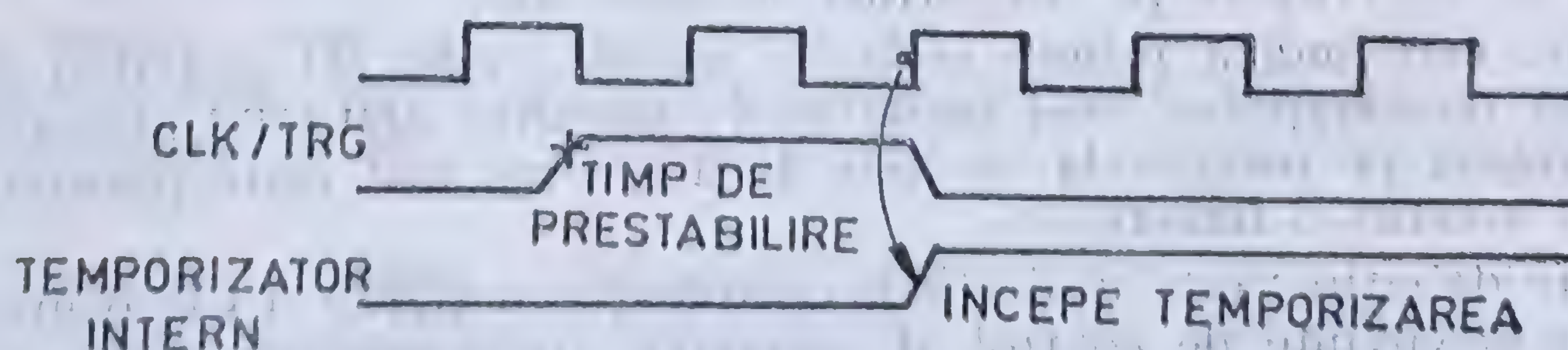


Fig. 4.8 Diagramă în timp pentru modul de temporizator.

RESET — intrare de inițializare; termină toate numărările în jos și desființează validarea întreruperilor, aducând la 0 biții de întrerupere în toate registrele de control; ieșirile ZC/TO și INT devin inactive; IEI=IEO, iar D0—D7 trec în starea de impedanță ridicată.

ZC/TO<sub>0</sub> — ZC/TO<sub>2</sub> — ieșiri de număr zero/timp expirat; trei pini ZC/TO, corespunzând canalelor 0, 1, 2 ale circuitului CTC (canalul 3 nu are acest pin) furnizează un impuls logic atît în modul de numărător cît și în cel de temporizator, cînd numărătorul descrescător al canalului a atins valoarea zero.

Un sistem tipic Z80 și încadrarea circuitului Z80 CTC în acest sistem, sînt ilustrate în figura 4.7.

#### Funcționarea în timp a circuitului Z80 CTC

În modul de temporizator, un impuls CLK/TRG declanșează temporizatorul (figura 4.8) pe al doilea front crescător care urmează, al semnalului de tact, CLK. Impulsul de declanșare este asincron și trebuie să aibă o lățime minimă. Un timp minim de 210 ns este necesar între frontul activ al lui CLK/TRG și următorul



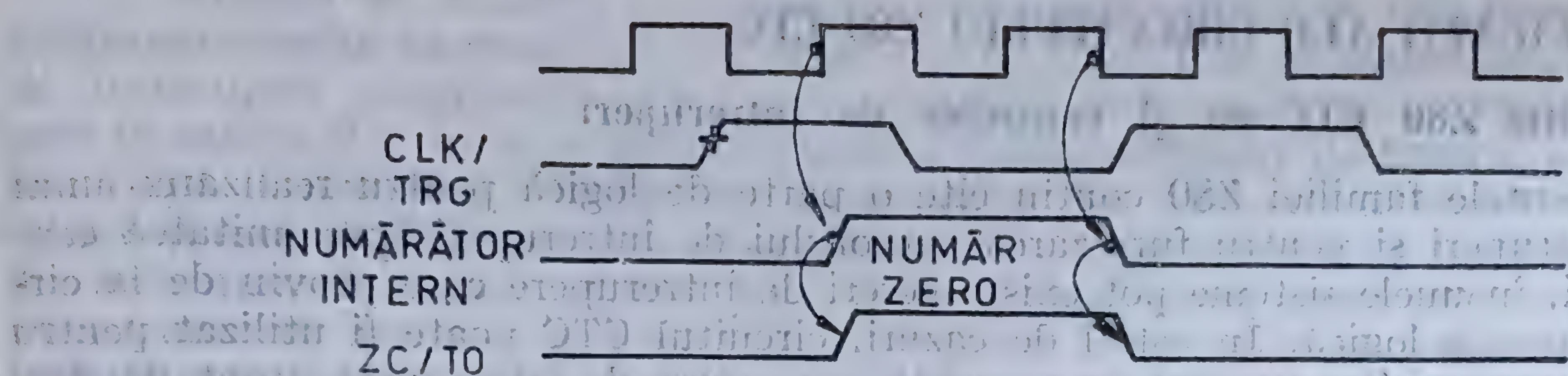


Fig. 4.9 Diagramă în timp pentru modul de numărător.

## Funcționarea la întreruperi

Circuitul Z80 CTC urmează protocolul la întreruperi al sistemului Z80, pentru întreruperi prioritare suprapuse și revenire din întrerupere, în care prioritatea unui periferic la întrerupere este determinată de poziția lui în lanțul de priorități. Semnalele IEI și IEO conectează circuitul în lanțul de priorități al sistemului. Dispozitivul cel mai apropiat de sursa de +5V are cea mai mare prioritate (fig. 4.10).

În interiorul circuitului CTC, prioritatea la întreruperi este determinată de numărul canalului, canalul 0 având cea mai mare prioritate, iar canalul 3, cea mai mică. Dacă un dispozitiv sau un canal este deservit într-o rutină de întrerupere, el nu poate fi întrerupt de un dispozitiv sau canal cu prioritate mai mică, pînă la terminarea servirii întreruperii. Dispozitive sau canale cu prioritate mai mare pot întrerupe servirea celor cu prioritate mai mică.

Un canal al circuitului Z80 CTC poate fi programat să ceară o întrerupere de fiecare dată cînd numărătorul descrescător al lui atinge valoarea 0. Unitatea centrală trebuie să fie programată pentru modul 2 de întreruperi. La un timp după cererea de întrerupere, unitatea centrală trimite un semnal de recunoaștere a întreruperii. Logica de control a întreruperilor din CTC determină canalul cu cea mai mare prioritate care a cerut o întrerupere. În continuare, dacă  $IEI = 1$ , arătînd că circuitul CTC are prioritate în cadrul lanțului de priorități, circuitul CTC plasează un vector de întrerupere de 8 biți pe magistrala de date a sistemului. Cei 5 biți de ordin superior ai acestui vector au fost înscriși în CTC în timpul procesului de programare. Următorii 2 biți sînt dați de logica de întreruperi din CTC, ca un cod binar care identifică cel mai prioritar canal care a cerut o întrerupere. Bitul cel mai puțin semnificativ este întotdeauna 0.

După apariția unei cereri de întrerupere, unitatea centrală trimite un semnal de recunoaștere a întreruperii ( $\overline{MI}$  și  $\overline{IORQ}$ ). Toate canalele sînt împiedicate să-și schimbe starea de cerere de întrerupere cînd  $\overline{MI}$  este activ (aproximativ cu două perioade de tact mai devreme ca  $\overline{IORQ}$ ). Semnalul  $\overline{RD}$  este la 1 pentru a distinge acest ciclu de cel de aducere a unei instrucțiuni din memorie.

Logica de întrerupere a circuitului CTC determină cel mai prioritar canal care a cerut o întrerupere. Dacă intrarea de validare a întreruperilor din CTC este  $IEI=1$ , cel mai prioritar canal din CTC care a cerut o întrerupere își plasează vectorul de întrerupere pe magistrala de date cînd  $\overline{IORQ}$  trece la 0. Două stări de așteptare sînt inserate automat în acest timp pentru a permite lanțului de priorități să se stabilizeze. Pot fi adăugate și stări de așteptare adiționale.

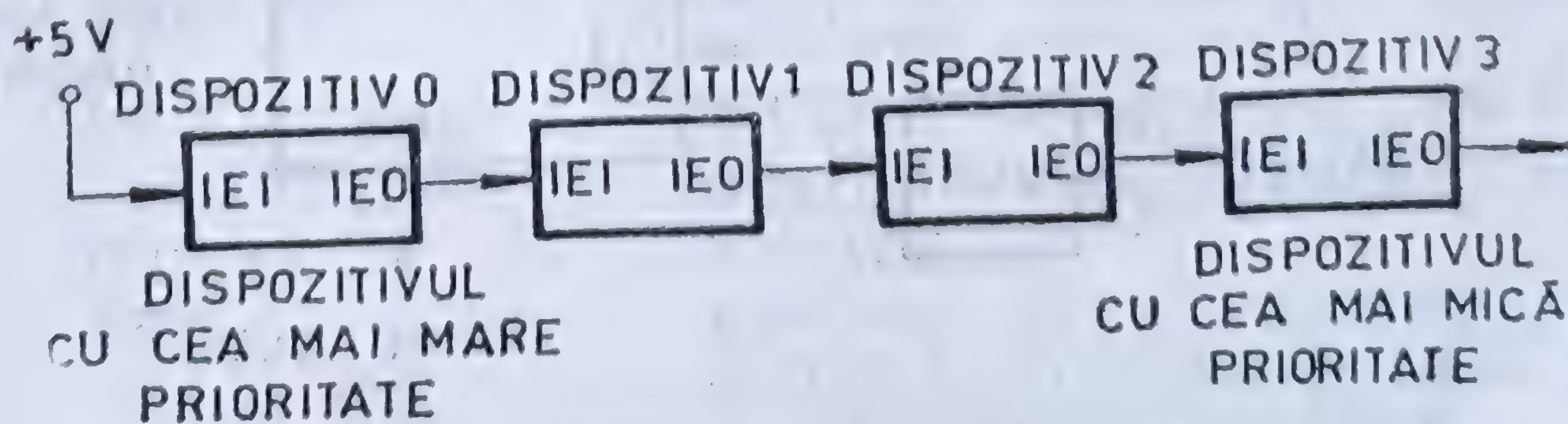


Fig. 4.10 Priorități la întrerupere într-un lanț de priorități.



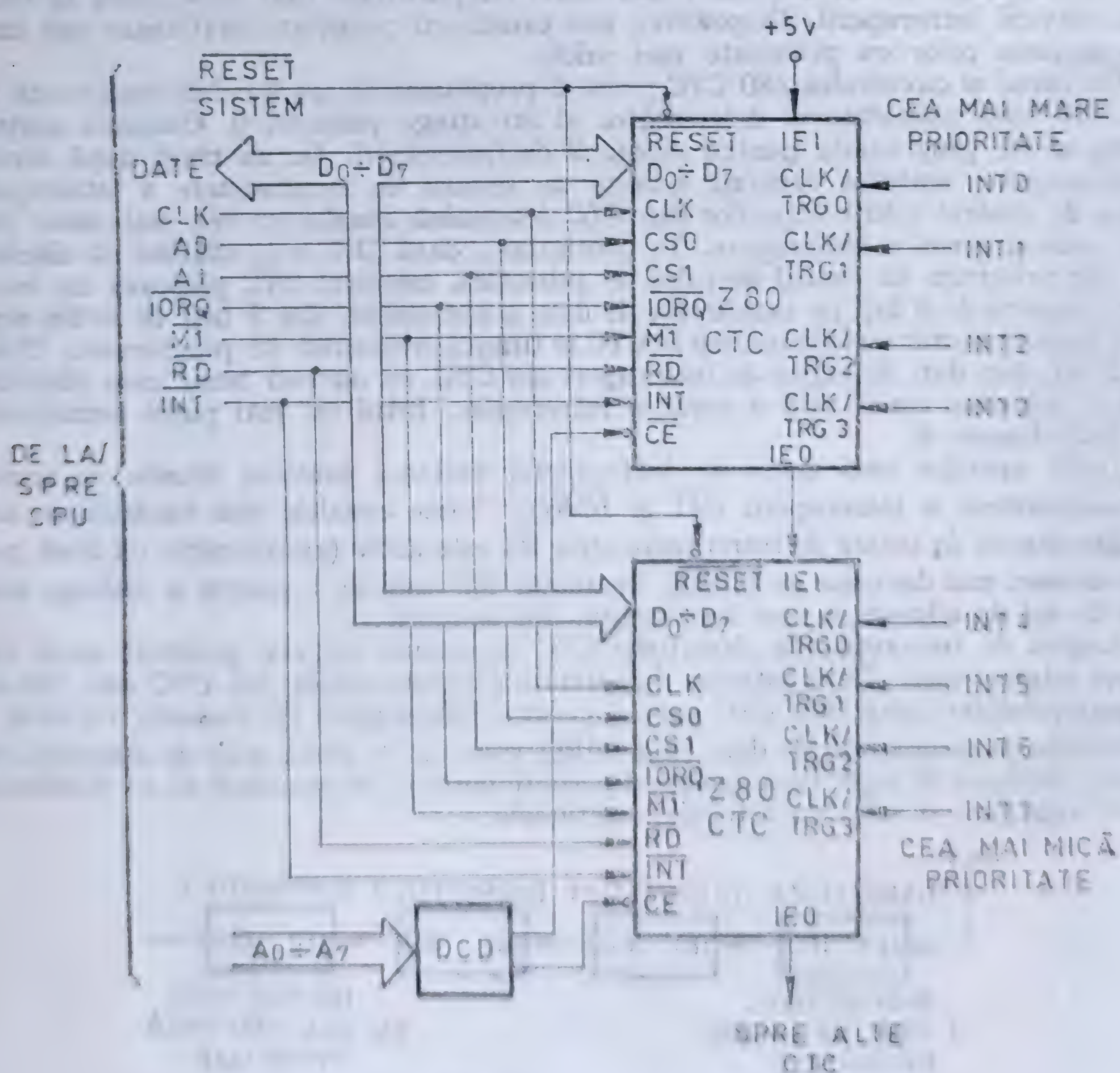
## 4.2. APLICATIILE CIRCUITULUI Z80 CTC

## 1. Circuitul Z80 CTC ca și controler de întreruperi

Componentele familiei Z80 conțin cîte o parte de logică pentru realizarea unui lanț de întreruperi și pentru furnizarea vectorului de întreruperi spre unitatea centrală. Totuși, în unele sisteme pot exista cereri de întrerupere care provin de la circuite fără această logică. În astfel de cazuri, circuitul CTC poate fi utilizat pentru a genera întreruperi într-un lanț de priorități, cu vector de întrerupere, front de declanșare, posibilitate de mascare și de temporizare, programabile.

Circuitele CTC pot fi conectate în cascadă, formînd un lanț cu maximum 256 intrări de întrerupere (fig. 4.11).

Fiecare canal din CTC poate genera o întrerupere după una sau mai multe (pînă la 256) tranziții, pe frontul ales activ, al semnalului de la intrare. Fiecare canal are propriul vector de întreruperi, care determină adresa la care se află adresa subrutinei de tratare a întreruperilor (obținută din vectorul de întreruperi combinat cu conținutul registrului I). Prioritatea cererilor de întrerupere este rezolvată de sistemul de priorități al CTC. Oricare canal poate fi „mascat”, dezactivînd cererile de întrerupere de la acest canal.



**Fig. 4.11** Circuitul CTC utilizat ca și controler de întreruperi.



O variantă de utilizare a unui canal CTC ca și intrare de întreruperi este de a programa canalul ca numărător cu constanta de timp 1 și cu frontul activ și vectorul de întreruperi programate. Când apare frontul programat, se generează o întrerupere în modul 2 și se va executa subrutina corespunzătoare de tratare a întreruperilor. După întrerupere, numărătorul descrescător din CTC este automat reîncărcat cu cifra 1 și canalul CTC așteaptă un alt front activ după executarea instrucțiunii RETI din subrutina de tratare a întreruperii. Deci, în timp ce un canal este în curs de servire, fronturile active pe intrarea CLK/TRG vor fi recunoscute doar după execuția instrucțiunii RETI. Alte canale ale CTC pot genera întreruperi care vor fi executate sau vor aștepta terminarea servirii întreruperii în curs, în funcție de prioritatea lor.

## 2. Utilizarea CTC pentru măsurarea duratei unui program

Durata execuției programelor se poate stabili însumând numărul de stări care corespunde instrucțiunilor. Pentru programe mai lungi, este mai comod să se utilizeze posibilitățile oferite de circuitul Z80 CTC, ca în exemplul listat în continuare. Prima parte reprezintă inițializarea circuitului. După inițializare, începe rularea programului a cărui durată se măsoară. În continuare, secțiunea finală citește și afișează valoarea înscrisă în canalul CTC, reprezentând durata programului. În ultima parte se listează un apel tipic. Programul de măsurare introduce o durată suplimentară de 59 de perioade de tact, valoare care trebuie scăzută din rezultatul citit în CTC.

### ETICHETA COD OPERAȚIE COMENTARIU

ENTRY :	; ADRESA DE REVENIRE IN MONITOR
WRITE: 2000H	; RUTINA DE SCRIERE LA CONSOLA
	; SE FIXEAZĂ CEA MAI MARE PERIOADA
	; DE TIMP IN CANALUL 3 CTC
CTCINI: PUSH AF	; ÎNȚĂLĂZĂ CTC; SALVEAZA AF
LD A, 27H	; CUVÎNT DE CONTROL AL CANALULUI:
	; DEZACTIVEAZA INTRERUPERILE;
	; TEMPORIZATOR; FACTOR PRESCALA-
	; RE 256; FRONT ACTIV CAZATOR; DE-
	; CLANSARE AUTOMATA; URMEA CON-
	; STANTA DE TIMP; INITIALIZARE PRIN
	; PROGRAM
OUT (CTCCH <sub>3</sub> ), A	; INSCRIE CUVINTUL DE CONTROL IN
	; CANALUL 3
XOR A	; STERGE ACUMULATORUL
OUT (CTCCH <sub>3</sub> ), A	; CONSTANTA DE TIMP 00H (256D) ESTE
	; INSCRISA IN CANALUL 3
POP AF	; REFACE AF
RET	; REVINE; INSTRUCȚIUNILE CARE UR-
	; MEAZA DUA DECLANSAREA TEMPO-
	; RIZATORULUI AU DURATA DE 20 PE-
	; RIOADE DE TACT
CTCIN: PUSH AF	; CITESTE CONTINUTUL NUMARATORU-
	; LUI CTC
IN A, (CTCCH <sub>3</sub> )	; CITESTE CONSTANTA DIN CANALUL 3
CALL WRITE	; SCRIE LA CONSOLA VALOAREA CITITA
POP AF	; REFACE AF
RET	; REVINE; INSTRUCȚIUNILE PINA LA
	; CITIREA NUMARATORULUI DUREAZA
	; 22 PERIOADE DE TACT



# EXEMPLU DE APEL TIPIC

```
CALL CTCINI ; APELEAZA SUBROUTINA DE INITIALI-
; ZARE CTC
; PROGRAMUL A CARUI DURATA SE MA-
; SOARA
CALL CTCIN ; APELEAZA SUBROUTINA DE MASURARE
; SI AFISARE
JP RENTRY ; SALT IN MONITOR
```

De notat că subrutina CTCINI introduce 20 de perioade de tact, iar CTCIN 22 de perioade, la care se adaugă 17 perioade pentru apelul lui CTCIN din programul de măsurat. Suma de 59 de perioade trebuie scăzută din rezultatul calculat. Dacă V este valoarea afișată la consolă, atunci numărul de decrementări ale canalului CTC este  $N = 256 - V$ , iar numărul de perioade de tact ale programului este  $256(256 - V) - 59$ , durată fiind  $\Delta t = [256 \cdot (256 - V) - 59] \cdot T_{CLK}$  ( $T_{CLK} = 0,4 \mu s$  pentru  $f_{CLK} = 2,5 \text{ MHz}$ ).

Programul măsurat nu trebuie să aibă mai mult de  $65536 - 59 = 65477$  perioade de tact.

## 3. Utilizarea CTC la generarea tactului de emisie-recepție pentru canalele circuitului SIO, într-un sistem Z80

Secțiunea de program descrisă permite determinarea vitezei de schimb a informației fixată la consola conectată la sistem, și fixarea aceleiași viteze în sistem, după recepția unui caracter de la consolă.

Programul monitor, înscris în memorii EPROM, de la adresa E000H, este atins după aplicarea semnalului RESET, forțând liniile A15, A14 și A13 de adresă la valoarea 1, cu ajutorul unui bistabil și a unor porți. Odată atinsă zona de memorie în care este înscris monitorul, se face un salt la adresa E003H, la care instrucțiunea IN PSYS desființează forțarea adreselor, realizată mai sus.

ADRE- SA	CODUL ETI- CHETA	COD OPERA- ȚIE	COMENTARIU
E000	C303E0	JMP RESET	; SALT LA E003H, CONTINUTUL PC ; FIIND EGAL CU ADRESA PE ME- ; MORIE, DUPA EXECUTARE
E003	DBF4	RE- IN/A,(PSYS) SET:	; CITESTE PSYS, STERGE FORTARE ; ADR
E005	2190FF	LD HL,SPUS	; ADRESA STIVA UTILIZATOR IN HL
E008	22E6FF	LD (ASP),HL	; DEPUNE SPUS LA ADRESA ASP
E00B	AF	XRA A	; STERGE A
E00C	320CFF	LD (ADR),A	; DEPUNE 0 LA FF0CH=ADR
E00F	3E05	LD A,05H	; CUVINT DE CONTROL CTC IN A
E011	D3D8	OUT(CTCCH0),A	; INSCRIE CUVINT DE CONTROL
E013	3E03	LD A,03H	; CUVINT DE CONTROL SIO IN A
E015	D3DE	OUT(SIOAC),A	; INSCRIE CUVINT DE CONTROL IN ; PORTUL A DIN SIO
E017	0EF5	LD C,RI	; ADRESA RI IN C; PE BITUL D7 ; SE VA CITI LINIA DE RECEPTIE
E019	110100	LD DE,0001H	; A DATELOR SERIALE
E01C	ED78	VI: IN A,(C)	; INCARCA 0001H IN DE
E01E	F21CE0	JP P,VI	; CITESTE LINIA DE RECEPTIE
E021	13	V2: INC DE	; SALT DACA NU ESTE BIT DE START ; INCREMENTEAZA DE DACA RI=1



E022	ED78	IN A,(C)	; CITESTE LINIA DE RECEPTIE
E024	FA21E0	JP M,V2	; SALT DACA LINIA SE MENTINE
			; LA 1
E027	314FE0	LD SP,TB-	; ADRESA TABEL, DIVIZARE CLK
		RATE1	
E02A	33	V3: INC SP	; SP INDICA NR. CICLURI PT.O
			; ANUMITA VITEZA
E02B	E1	POP HL	; NUMARUL DE CICLURI IN HL
E02C	37	SCF	; 1 IN BITUL DE TRANSPORT
E02D	ED52	SBC HL,DE	; NR. CICLURI TABEL - NR. CICLU-
			; RI MASURAT - 1 IN HL.
E02F	38F9	JR C,V3	; PENTRU (HL) < 0, SALT LA V3,
			; COMPARA CU NR. CICLURI PT.
			; VITEZA IMEDIAT INFERIOARA
E031	3B	DEC SP	; DECREMENTEAZA DACA (HL) ≥ 0
E032	F1	POP AF	; FACTOR DIVIZARE PENTRU CTC-
			; CH0
E033	D3D8	OUT(CTCCH0),	; INSCRIE FACTORUL DIN A IN CTC
		A	

#### TABEL PENTRU CALCULAREA VITEZEI DE SCHIMB A INFORMATIEI (BAUD RATE) CU CONSOLA

E04E	E2	TBRA-	; OCTET FARA SEMNIFICATIE
		TE1:	
E04F	0E00	TBRA-	; NR. CICLURI PT. 9600BD (13D)
		TE:	
E051	01		; FACTOR DIVIZARE PT. 9600BD (1D)
E052	1A00		; NR. CICLURI PT. 4800BD (26D)
E054	02		; FACTOR DIVIZARE PT. 4800BD (2D)
E055	3400		; NR. CICLURI PT. 2400BD (52D)
E057	04		; FACTOR DIVIZARE PT. 2400BD (4D)
E058	6800		; NR. CICLURI PT. 1200BD (104D)
E05A	08		; FACTOR DIVIZARE PT. 1200BD (8D)
E05B	D000		; NR. CICLURI PT. 600BD (208D)
E05D	10		; FACTOR DIVIZARE PT. 600BD (16D)
E05E	A001		; NR. CICLURI PT. 300BD (416D)
E060	20		; FACTOR DIVIZARE PT. 300BD (32D)
E061	FF7F		; NR. CICLURI PT. 110BD (32767D)
E063	57		; FACTOR DIVIZARE PT. 110BD (87D)

Stiva utilizatorului se fixează la adresa SPUS=FF90H din RAM. Secțiunea de program listată fixează frecvența de schimb a informației, pe care circuitul CTC, canalul 0 o aplică pe intrările de tact ale circuitului SIO, canalul A, divizând frecvența de tact a sistemului cu o valoare care se determină în funcție de durata bitului de start trimis de la consolă spre sistem, după aplicarea semnalului RESET. Caracterul trimis trebuie să aibă bitul D0, care urmează după bitul de start (cu valoarea 0 logic), la valoarea D0 = 1, Un caracter de acest tip poate fi CAR RETURN (cod ASCII „OD”).

După înscrierea cuvintelor de control în CTC, canalul 0 și SIO, canalul A, se măsoară durata bitului de start, în număr de cicluri, între adresele V2-E021H și E024H, parcurse dacă bitul de pe linia de recepție de date RI (citită la adresa F5H, în care bitul 7 conține valoarea logică a liniei de recepție de date seriale) este la 1 logic. O inversare hard a liniei de recepție face ca bitul de start să fie citit cu valoarea logică 1.



Durata unui ciclu este de 30 de stări, deci  $30 \times 0,4 \mu s = 12 \mu s$  la frecvența de tact CLK = 2,5 MHz. Durata bitului de start la viteze de schimb diferite, alături de numărul de cicluri din tabel și factorul de divizare pentru fiecare viteză se listează în continuare:

Factor divizare CLK cu CTC canalul 0	Viteza (Bd)	Bit de start ( $\mu s$ )	Număr de cicluri măsurat	Număr de cicluri din tabel
87	110	9090	758	32767
32	300	3333	278	416
16	600	1666	139	208
8	1200	833	70	104
4	2400	416	35	52
2	4800	208	18	26
1	9600	104	9	13

După determinarea duratei bitului de start, în număr de cicluri măsurate, conținut în registrul dublu DE, se încarcă un număr de cicluri din tabel în HL și se face scăderea (HL) - (DE) - 1. Dacă nu apare transport, numărul de cicluri din tabel corespunde vitezei de schimb a informației cu care s-a transmis bitul de start. Dacă apare transport, se trece la constanta „număr de cicluri” mai mare, care corespunde vitezei imediat inferioare. Se observă că numărul de cicluri din tabel pentru cele 5 viteze superioare s-a ales ca media aritmetică a numerelor de cicluri măsurate pentru viteza aleasă și pentru cea imediat inferioară. De exemplu pentru 9600Bd, constanta din tabel nu este 9, ci  $\left[ \frac{9+18}{2} \right] = 13$  „cicluri”. Pentru 300Bd s-a ales o constantă între numerele de cicluri măsurate pentru 300Bd și 110Bd, iar pentru 110Bd, o constantă superioară celei măsurate.

Dacă s-a identificat viteza, instrucțiunea POP AF introduce constanta de divizare pentru canalul 0 din CTC, în acumulator, iar următoarea instrucțiune înscrie

constantă (de timp în canalul 0, determinând frecvența de tact corectă pentru circuitul SIO.

De notat că la programarea canalului 0 CTC, semnificația cuvântului de control este: fără întreruperi, front căzător, mod de temporizator, factor de prescalare, și urmează constanta de timp.

Schema utilizată pentru măsurarea vitezei de schimb a informației seriale cu consola este prezentată în fig. 4.12.

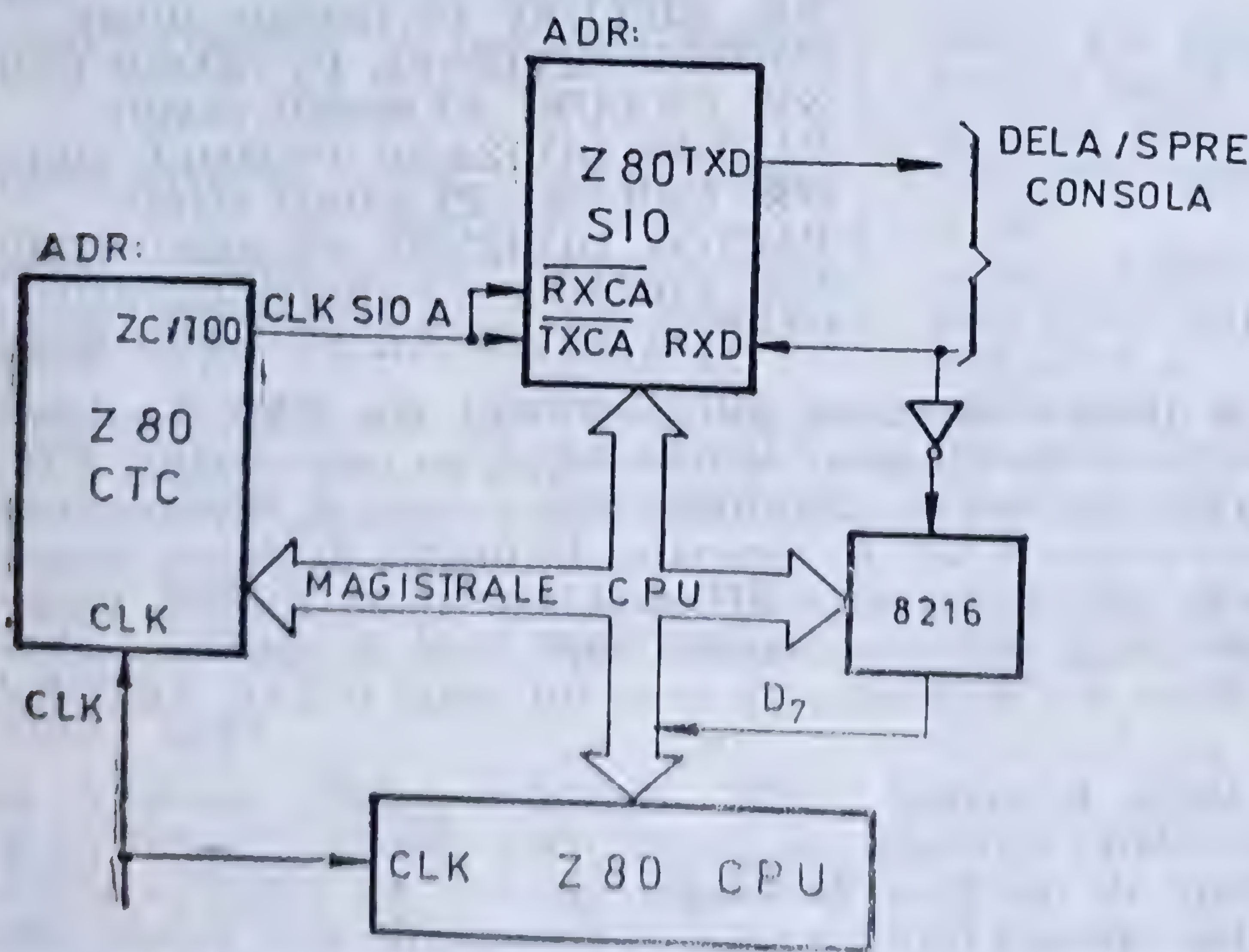


Fig. 4.12 Măsurarea vitezei de schimb a informației seriale, cu ajutorul circuitului Z80 CTC.



#### 4. Utilizarea circuitului CTC pentru execuția pas cu pas a unui program

O posibilitate de a realiza rularea pas cu pas, „cite o instrucțiune”, a unui program, constă în utilizarea întreruperilor NMI, generate cu ajutorul unui circuit CTC. Un canal al circuitului are aplicat semnalul  $\overline{M1}$  pe intrarea CLK/TRG, iar ieșirea ZC/TO este conectată la intrarea  $\overline{NMI}$  a microprocesorului (fig. 4.13).

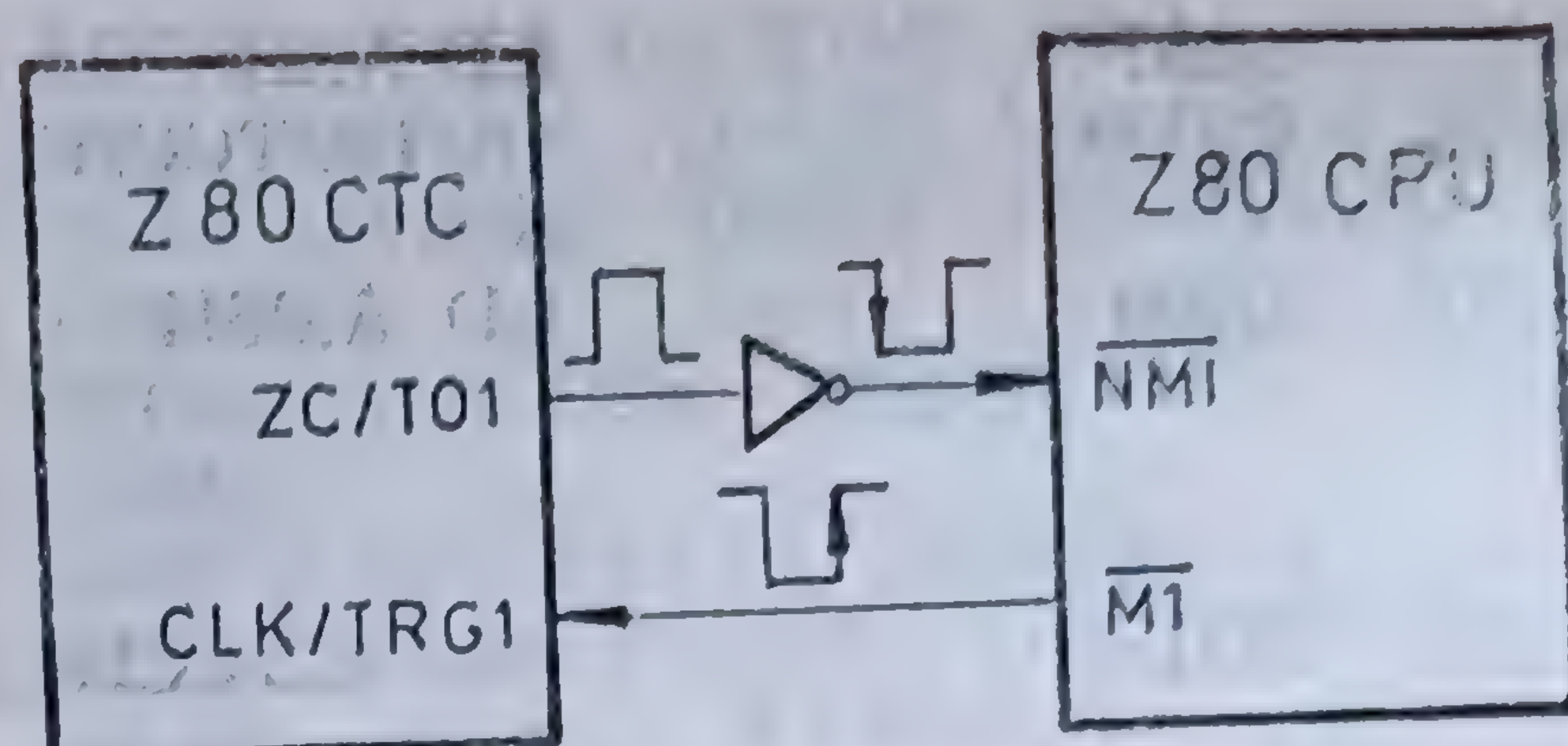


Fig. 4.13 Conexiuni Z80 CTC-Z80 CPU pentru rularea pas cu pas a unui program.

Canalul este programat ca numărător cu o astfel de constantă încât să genereze un impuls la ieșirea ZC/TO atunci când este adus din memorie codul operației instrucțiunii care se execută în regim pas cu pas. Decrementarea numărătorului se execută la fiecare impuls  $\overline{M1}$ , deci în primul ciclu de mașină al fiecărei instrucțiuni. După executarea unei instrucțiuni din programul care se rulează pas cu pas, se afișează la consolă, pentru simplitate, numai conținutul acumulatorului și adresa următoarei instrucțiuni care se va executa. La început se afișează și adresa primei instrucțiuni a programului, astfel încât pe o linie apare adresa instrucțiunii care s-a executat și conținutul acumulatorului după executarea ei. Programul se execută începând cu adresa PP=FC00H. La adresa FD00H programul va găsi, începând cu octetul inferior, adresa de început a programului de rulat. La adresa 0066H, la care se transferă execuția programului după primirea cererii  $\overline{NMI}$  se insercază instrucțiunea „JP PP1”, pentru afișarea conținutului acumulatorului și a adresei următoarei instrucțiuni.

Semnificația cuvântului de comandă pentru canalul CTC, 57H = 01010111B este: fără validare de întreruperi  $\overline{INT}$ , mod de numărător, front activ crescător, urmează constanta de timp și inițializare prin program. Programul poate fi completat pentru afișarea conținutului tuturor registrelor microprocesorului, a numelor acestora, precum și pentru execuția mai multor instrucțiuni la un pas. În varianta prezentată, fiecare instrucțiune a programului urmărit, cu excepția primei instrucțiuni, se execută după primirea unui caracter (oarecare) de la consolă. Prima instrucțiune se execută când se lansează programul de la adresa PP. Programul este listat în continuare:

ADRE- SA	CODUL	ETI- CHETA	COD. OPERA- ȚIE	COMENTARIU
0066	C324FC	RST:	JP PP1	; SALT LA PP1 PT.AFISARE A SI ; ADRESA; SE AJUNGE AICI DUPA ; EXECUTAREA UNEI INSTRUCȚI- ; UNI DIN PROGRAMUL URMARIT ; ADRESA PROGRAMULUI DE ; EXECUTAT PAS CU PAS ; INCARCA 00H, SELECTEAZA CANA- ; LUL, CONSOLA ; TRECE LA RIND NOU
FD00	XXXX		ADR PROG	
FC00	1E00	PP:	LD E,00H	; OCTET SUPERIOR ADRESA IN A ; IMPRIMA OCTETUL SUPERIOR ; OCTET INFERIOR ADRESA IN A ; IMPRIMA OCTETUL INFERIOR ; IMPRIMA UN SPATIU ; ADRESA PROGRAM DE RULAT IN ; HI,
FC03	CD9CE5		CALL CRLF	
FC05	3A01FD		LD A,FD01H	
FC08	CD8BE5		CALL PACC	
FC0B	3A00FD		LD A,FD00H	
FC0E	CD8BE5		CALL PACC	
FC11	CDA5E5		CALL SPACE	
FC14	2A00FD		LD HL,FD00H	



FC17	3E57	LD A,CDCH1	; CUVINT COMANDA CANAL CTC IN A
FC19	D3D9	OUT(CTCCH1),	; PROGRAMEAZA CANALUL 1 CTC
		A	
FC1B	3E06	LD A,06H	; CONSTANTA DE TIMP IN A; CERE-
			; REA NMI VA APARE LA A 6-A
			; INSTRUCȚIUNE DUPA PROGRAMA-
			; REA CONSTANTEI
FC1D	D3D9	OUT(CTCCH1),	; PROGRAMEAZA CONSTANTA DE
		A	; TIMP
FC1F	00	NOP	; NICI O OPERATIE, PENTRU A PER-
FC20	00	NOP	; MITE O CONSTANTA DE TIMP MAI
FC21	00	NOP	; MARE CU 4; PERMITE ASTFEL
FC22	00	NOP	; INITIALIZAREA CANALULUI CTC
			; INAINTE DE EMITEREA UNEI
			; NOI CERERI NMI
FC23	E9	JP (HL)	; SALT LA PRIMA INSTRUCȚIUNE
			; A PROGRAMULUI DE EXECUTAT,
			; UNDE SE VA EMITE O CERERE
			; NMI
FC24	F5	PP1: PUSH AF	; SALVEAZA AF
FC25	3E57	LD A,CDCH1	; CUVINT DE COMANDA CANAL CTC
FC27	D3D9	OUT(CTCCH1),	; PROGRAMEAZA CANALUL 1 CTC;
		A	; NUMARAREA ESTE OPRITA
FC29	D9	EXX	; INTERSCHIMBA REGISTRELE BC,
			; DE,HL,CU BC,DE,HL
FC2A	1E00	LD E,00	; INCARCA 00, SELECTEAZA CONSO-
			; LA
FC2C	F1	POP AF	; REFACE AF
FC2D	F5	PUSH AF	; SALVEAZA AF SI REFACE SP
FC2E	CD8BE5	CALL PACC	; IMPRIMA A DUPA EXECUTAREA
			; INSTRUCȚIUNII
FC31	CD9CE5	CALL CRLF	; TRECE LA RIND NOU
FC34	33	INC SP	; INCREMENTEAZA SP CU 2 PT.A
FC35	33	INC SP	; INDICA LOCATIA IN CARE S-A
			; DEPUȘ ADRESA URMATOAREI IN-
			; STRUCȚIUNI, DUPA CEREREA NMI
FC36	C1	POP BC	; ADRESA DE MAI SUS IN BC
FC37	3B	DEC SP	; DECREMENTEAZA SP DE 4 ORI PT.
FC38	3B	DEC SP	; A INDICA LOCATIA DE MEMORIE
FC39	3B	DEC SP	; IN CARE S-A DEPUȘ CONTINUTUL
FC3A	3B	DEC SP	; LUI A DIN GRUPUL AF
FC3B	C5	PUSH BC	; SALVEAZA ADRESA URMATOAREI
			; INSTRUCȚIUNI
FC3C	78	LD A,B	; OCTET SUPERIOR ADRESA IN A
FC3D	CD8BE5	CALL PACC	; IMPRIMA OCTET SUPERIOR ADRE-
			; SA
FC40	C1	POP BC	; REFACE ADRESA
FC41	79	LD A,C	; OCTET INFERIOR ADRESA IN A
FC42	CD8BE5	CALL PACC	; IMPRIMA OCTET INFERIOR ADRE-
			; SA
FC45	CDA5E5	CALL SPACE	; IMPRIMA UN SPATIU
FC48	CD22E5	CALL RDCHR	; AȘTEAPȚA UN CARACTER DE LA
			; CONSOLA
FC4B	F1	POP AF	; REFACE AF
FC4C	D9	EXX	; REFACE BC,DE,HL



FC4D	3E06	LD A,06H	; CONSTANTA DE TIMP IN A ; CERE-
			; REA NMI VA APARE LA A 6-A IN-
			; STRUCȚIUNE DUPA PROGRAMA-
			; REA EI.
FC4F	D3D9	OUT(CTCH1),	; PROGRAMEAZA CONSTANTA DE
		A	; TIMP
FC51	00	NOP	; NICI O OPERATIE, PENTRU A PER-
FC52	00	NOP	; MITE O CONSTANTA DE TIMP MAI
FC53	00	NOP	; MARE CU 4; PERMITE ASTFEL
FC54	00	NOP	; OPRIREA DIN NUMARARE DUPA
			; APARITIA UNEI CERERI NMI
FC55	ED45	RETN	; REVINE DIN INȚRERUPEREA NE-
			; MASCABILA, LA URMATOAREA IN-
			; STRUCȚIUNE DIN PROGRAMUL DE
			; RULAT PAS CU PAS, UNDE SE VA
			; EMITE O NOUA CERERE DE INȚRE-
			; RUPERE NMI



## CAPITOLUL V

### CIRCUITUL DE CONTROL PENTRU INTRARE-IEȘIRE SERIALĂ, Z80 SIO

#### 5.1. DESCRIEREA CIRCUITULUI DE CONTROL PENTRU INTRARE-IEȘIRE SERIALĂ

Circuitul conține 2 canale independente, full-duplex, cu linii de stare și control pentru modem-uri sau alte dispozitive. Vitezele de schimb ale informației sînt cuprinse în gama  $0 \div 500$  kbit/sec, pentru un semnal de tact de 2,5 MHz (Z80 SIO) sau de  $0 \div 800$  kbit/sec, pentru 4 MHz (Z80A SIO). În modul asincron, se asigură un protocol complet pentru a transmite caractere de 5, 6, 7 sau 8 biți.

Se pot introduce biți de stop în număr variabil, diferite divizări ale semnalului de tact, generări/detectări de pauză (break), paritate, sau ieșiri din cadru.

În modul sincron, se pot transmite mesaje de tip bit sau octet, cu 5, 6, 7 sau 8 biți/caracter, în standardele IBM Bisync, SDLC, HDLC, CCITT-X.25, sau altele. Generarea și căutarea automată de caractere de redondanță ciclică (CRC), de caractere de sincronizare, introducerea/ștergerea de zerouri, ca și introducerea de indicatori sînt de asemenea posibile.

Registrele de recepție a datelor au 4 tampoane, iar cele de transmisie 2 tampoane. Întreruperile sînt tratate, fără logică externă, într-un lanț de întreruperi, cu utilizarea vectorilor de întrerupere.

#### Descrierea conexiunilor circuitului Z80 SIO

Figura 5.1 prezintă funcțiile logice ale circuitului Z80 SIO—0. Limitarea la 40 de conexiuni face imposibilă existența separată a semnalelor de tact recepție ( $\overline{RXC}$ ), tact transmisie ( $\overline{TXC}$ ), terminal de date gata ( $\overline{DTR}$ ) și sincronizare ( $\overline{SYNC}$ ), pentru ambele canale. Ca urmare, pentru canalul B, fie lipsește un semnal, fie două intrări sînt legate împreună: varianta Z80 SIO—0 are toate cele 4 semnale, dar  $\overline{TXCB}$  și  $\overline{RXCB}$  sînt conectate intern; varianta Z80 SIO—1 are lipsă semnalul  $\overline{DTRB}$ ; varianta Z80 SIO—2 are lipsă semnalul  $\overline{SYNCB}$ . Ultima versiune este cel mai adesea preferată în aplicații. Semnalele pentru oricare dintre variante sînt descrise în continuare.

$B/\overline{A}$  — intrare pentru selectarea canalului A (cu 0 logic) sau a canalului B (cu 1 logic); linia A0 de adresă de la unitatea centrală este utilizată în mod obișnuit pentru această selecție.

$C/\overline{D}$  — intrare de selecție pentru control sau date; definește tipul informației transferate între unitatea centrală și circuitul SIO: 1 logic pe această linie în timpul unei operații de scriere face ca informația de pe magistrala de date să fie interpretată ca o comandă pentru canalul selectat de linia  $B/\overline{A}$ ; un 0 logic arată că informația constituie o dată; linia de adresă A1 este în mod obișnuit conectată la această intrare.

$\overline{CE}$  — intrare de selecție a circuitului; un 0 logic face ca SIO să accepte comenzi sau date de la CPU în timpul unui ciclu de scriere sau să transmită date spre CPU în timpul unui ciclu de citire.

CLK — intrare de tact; se aplică semnalul de tact cu o singură fază al sistemului Z80, pentru sincronizarea semnalelor interne.

$\overline{CTSA}$ ,  $\overline{CTSB}$  — intrări care indică starea de "gata pentru transmitere" a canalului; cînd sînt programate ca semnale de auto-validare, un 0 pe aceste intrări



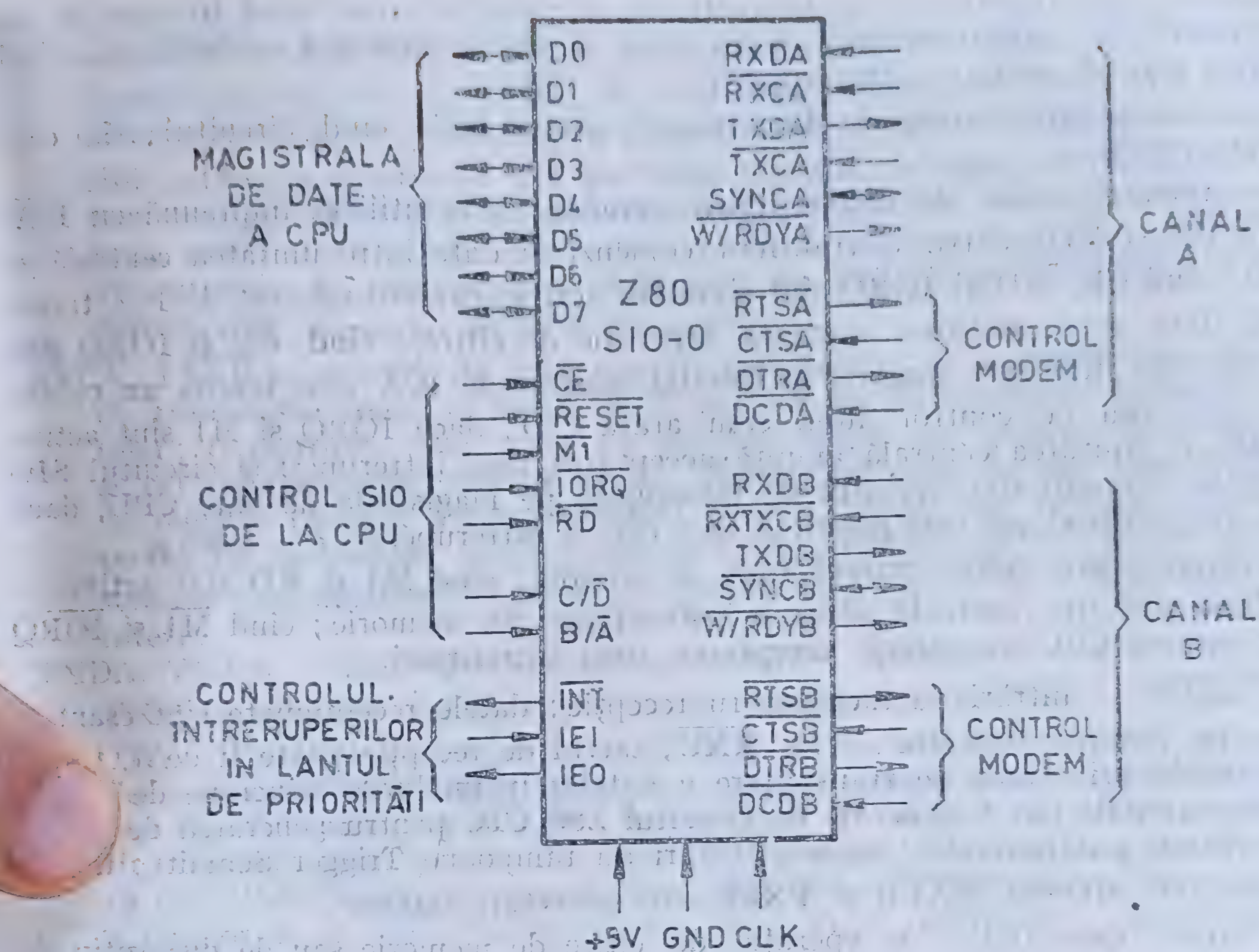


Fig. 5.1 Funcțiile logice ale circuitului Z80 SIO-0.

validează transmitătorul respectiv; dacă nu sînt programate ca auto-validări, aceste intrări pot fi programate ca intrări de uz general; ambele au tamponare Trigger-Schmitt, pentru a detecta semnalele cu creștere lentă; circuitul SIO detectează impulsurile pe aceste intrări și întrerupe unitatea centrală pe ambele tipuri de tranziții logice. Tamponarele Trigger-Schmitt nu garantează o margine specificată a nivelului de zgomot.

D0-D7 — Magistrala de date bidirecțională, cu 3 stări, a sistemului; transferă date și comenzi între unitatea centrală și circuitul Z80 SIO; D0 este cel mai puțin semnificativ bit.

DCDA, DCDB — intrări pentru detecția purtătoarei de date; aceste semnale funcționează ca intrări de validare a receptorului, dacă SIO este programat pentru auto-validare; dacă nu, intrările pot fi folosite ca intrări de uz general, cu tamponare Trigger-Schmitt, pentru a permite și aplicarea de semnale cu variație lentă; circuitul SIO detectează impulsurile pe aceste linii și întrerupe unitatea centrală pe ambele tipuri de tranziții logice;

DTRA, DTRB — ieșiri care indică starea de "gata" a terminalului de date; aceste ieșiri urmăresc starea programată în circuitul Z80 SIO; ele pot fi programate și ca ieșiri de uz general; în varianta Z80 SIO-1, nu există semnalul DTRB.

IEI — intrare de validare a întreruperilor; este utilizată împreună cu semnalul IEO, pentru a forma un lanț de priorități, cînd există mai multe dispozitive controlate prin întreruperi; un 1 logic arată că nici un dispozitiv cu prioritate mai mare nu este servit de unitatea centrală într-o rutină de servire a întreruperii.

IEO — ieșire de validare a întreruperilor; este la 1 logic numai dacă IEI=1 și unitatea centrală nu deservește nici o întrerupere de la acest SIO; semnalul



blochează întreruperea de la dispozitivele cu prioritate mai mică în timp ce un dispozitiv cu prioritate mai mare este servit de unitatea centrală în cadrul rutinei sale de servire a întreruperii.

$\overline{INT}$  — ieșire pentru cerere de întrerupere; este 0 logic când circuitul SIO cere o întrerupere.

$\overline{IORQ}$  — intrare; cerere de intrare/ieșire; semnalul este utilizat împreună cu  $B/\overline{A}$ ,  $\overline{C/D}$ ,  $\overline{CE}$  și  $\overline{RD}$  pentru a transfera comenzi și date între unitatea centrală și SIO; când  $\overline{CE}$ ,  $\overline{RD}$  și  $\overline{IORQ}$  sînt simultan active, canalul selectat de  $B/\overline{A}$  transferă date spre unitatea centrală (operație de citire); când  $\overline{CE}$  și  $\overline{IORQ}$  sînt active, dar  $\overline{RD}$  este inactiv, în canalul selectat de  $B/\overline{A}$  este înscris un cuvînt de date sau de control după cum arată  $\overline{C/D}$ ; dacă  $\overline{IORQ}$  și  $\overline{MI}$  sînt active simultan, unitatea centrală anunță acceptarea unei întreruperi și circuitul SIO își plasează automat vectorul de întreruperi pe magistrala de date CPU, dacă este dispozitivul cel mai prioritar care cere o întrerupere.

$\overline{MI}$  — intrare, care indică primul ciclu de mașină; când  $\overline{MI}$  și  $\overline{RD}$  sînt active simultan, unitatea centrală aduce o instrucțiune din memorie; când  $\overline{MI}$  și  $\overline{IORQ}$  sînt active, CPU comunică acceptarea unei întreruperi.

$\overline{RXCA}$ ,  $\overline{RXCB}$  — intrări de tact pentru recepție; datele recepționate sînt eșantionate pe frontul crescător al lui  $\overline{RXC}$ ; tactul de recepție poate fi de 1, 16, 32 sau 64 de ori viteza de transmitere a datelor în modurile asincrone de lucru; aceste semnale pot fi generate de circuitul Z80 CTC pentru generarea de viteze de schimb programabile; ambele intrări au tampoane Trigger Schmitt; în varianta Z80 SIO-0,  $\overline{RXCB}$  și  $\overline{TXCB}$  sînt conectate intern.

$\overline{RD}$  — intrare care indică o operație de citire de memorie sau de dispozitiv de I/E, dacă este la 0 logic; se utilizează în combinație cu semnalele  $B/\overline{A}$ ,  $\overline{CE}$  și  $\overline{IORQ}$  pentru a transfera date de la SIO la unitatea centrală.

$\overline{RXDA}$ ,  $\overline{RXDB}$  — intrări pentru datele recepționate serial, la nivel TTL.

$\overline{RESET}$  — intrare de inițializare; un 0 logic dezactivează ambii receptori și transmițători, forțează  $\overline{TXDA}$  și  $\overline{TXDB}$  la 1 logic (nivel de marcă), forțează semnalele de control ale modemului la 1 și dezactivează toate întreruperile; registrele de control trebuie înscrise după ce SIO este inițializat, înainte de a transmite sau recepționa date.

$\overline{RTSA}$ ,  $\overline{RTSB}$  — ieșiri, pentru cerere de transmitere; când bitul RTS din registrul de scriere 5 este înscris cu 1, ieșirea  $\overline{RTS}$  trece la valoarea 0; când bitul RTS este 0 în modul asincron, ieșirea trece la 1 după ce transmițătorul este gol; în modurile sincrone pinul  $\overline{RTS}$  urmărește starea bitului 5 din registrul de scriere 5; ambele semnale pot fi utilizate ca ieșiri de uz general.

$\overline{SYNCA}$ ,  $\overline{SYNCB}$  — intrări de sincronizare; aceste semnale pot funcționa fie ca intrări, fie ca ieșiri; în modul de recepție asincronă, sînt intrări similare cu  $\overline{CTS}$  și  $\overline{DCD}$ ; în acest mod tranzițiile pe aceste linii afectează starea biților Sync/Hunt în registrul de citire 0, fără a avea altă funcție; în modul de sincronizare externă, aceste linii funcționează și ca intrări; când se realizează sincronizarea externă, semnalul  $\overline{SYNC}$  trebuie adus la 0 pe al doilea front crescător al lui  $\overline{RXC}$ , după acel front al lui  $\overline{RXC}$  pe care s-a primit ultimul bit al caracterului de sincronizare; deci, după detectarea caracterului de sincronizare, logica externă trebuie să aștepte două perioade complete de tact de recepție pentru a activa intrarea  $\overline{SYNC}$ ; odată adusă la 0, trebuie menținută astfel pînă cînd unitatea centrală informează logica externă de detecție a sincronizării că s-a pierdut sincronizarea sau că urmează să înceapă un nou mesaj;



asamblarea caracterelor începe pe frontul crescător al lui  $\overline{RXC}$ , care precede frontul căzător al lui  $\overline{SYNC}$ , în modul de sincronizare externă; în modul de sincronizare internă (Monosync și Bisync) aceste semnale funcționează ca ieșiri care sînt active în intervalul din perioada tactului de recepție ( $\overline{RXC}$ ) în care sînt recunoscute caracterele de sincronizare; condiția de sincronizare nu este captată, astfel încît aceste ieșiri sînt active de fiecare dată cînd este recunoscut un caracter de sincronizare, indiferent de limitele caracterului; în versiunea Z80 SIO-2, semnalul  $\overline{SYNCB}$  nu există.

$\overline{TXCA}$ ,  $\overline{TXCB}$  — intrări de tact pentru transmisie; în modurile asincrone, tactul de transmisie poate fi de 1, 16, 32 sau 64 de ori viteza de transmitere a datelor; o restricție este aceea că multiplicatorul ales pentru tactul de recepție și cel de transmisie trebuie să fie același; intrările de tact pentru transmisie au registre tampon Trigger Schmitt, pentru creșteri sau scăderi lente de nivel; tactul de transmisie poate fi generat de circuitul Z80 CTC, pentru viteze de schimb programabile; în varianta Z80 SIO-0,  $\overline{TXCB}$  și  $\overline{RXCB}$  sînt conectate intern.

$\overline{TXDA}$ ,  $\overline{TXDB}$  — ieșiri de transmitere serială a datelor, la nivel TTL; semnalul  $\overline{TXD}$  își schimbă valoarea pe frontul căzător al lui  $\overline{TXC}$ .

$\overline{W/RDYA}$ ,  $\overline{W/RDYB}$  — ieșiri, cu drenă în gol; cînd sînt programate pentru funcția de așteptare (wait), sînt aduse la 1 logic; programate pentru funcția "gata" (ready) sînt aduse la 0 logic; aceste ieșiri cu dublu rol pot fi programate ca linii Ready (gata) pentru un controler de acces direct la memorie (DMA), sau ca linii Wait (așteaptă) care sincronizează unitatea centrală cu viteza de schimb a datelor de la SIO; starea acestor ieșiri după inițializarea circuitului este cu drenă în gol.

### Descrierea funcțională

Diagrama bloc a circuitului Z80 SIO este prezentată în fig. 5.2. Posibilitățile funcționale ale circuitului pot fi descrise din două puncte de vedere diferite: ca un dispozitiv de comunicare de date, el transmite și recepționează date seriale într-o varietate de moduri de comunicare de date (prin mod se înțelege un protocol de comunicare de date); ca dispozitiv periferic al familiei Z80, interacționează cu unitatea centrală și cu alte dispozitive periferice, utilizînd magistralele de date, adrese și control în comun cu ele, în cadrul structurii Z80 de priorități la întrerupere; ca periferic pentru alte microprocesoare, circuitul SIO oferă posibilități ca întreruperi fără vector de întrerupere, căutare repetată sau explorare, și conversație simplă.

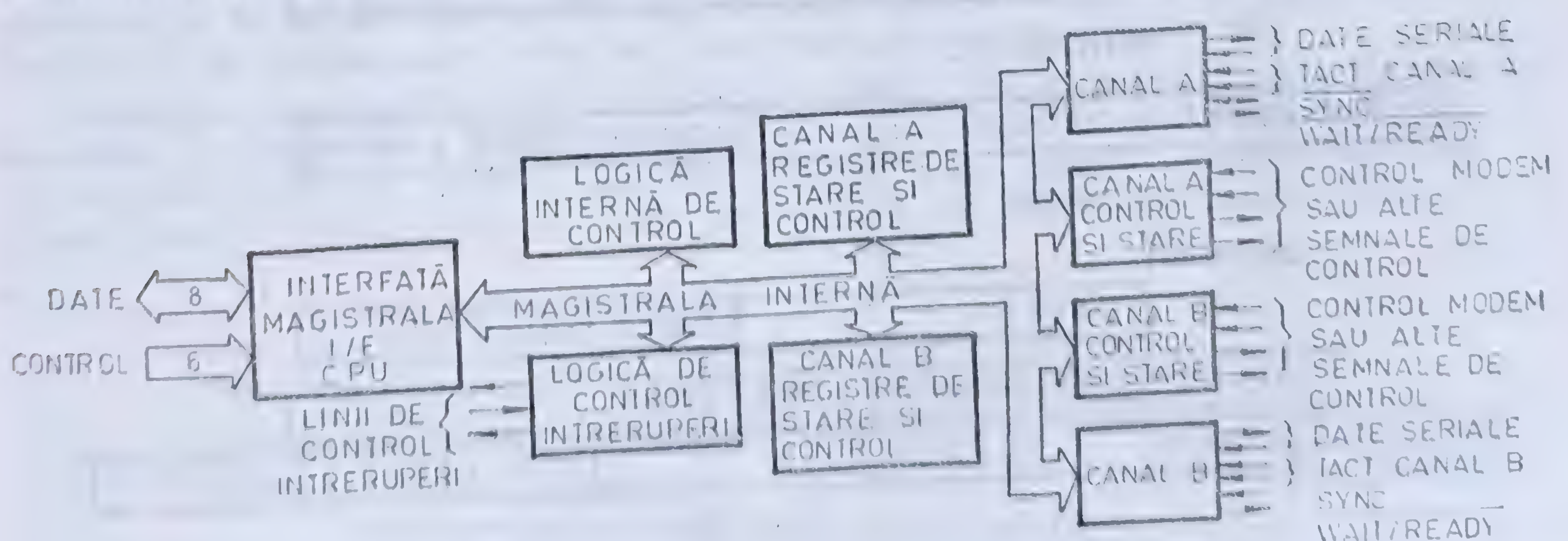


Fig. 5.2 Diagrama bloc a circuitului Z80 SIO.



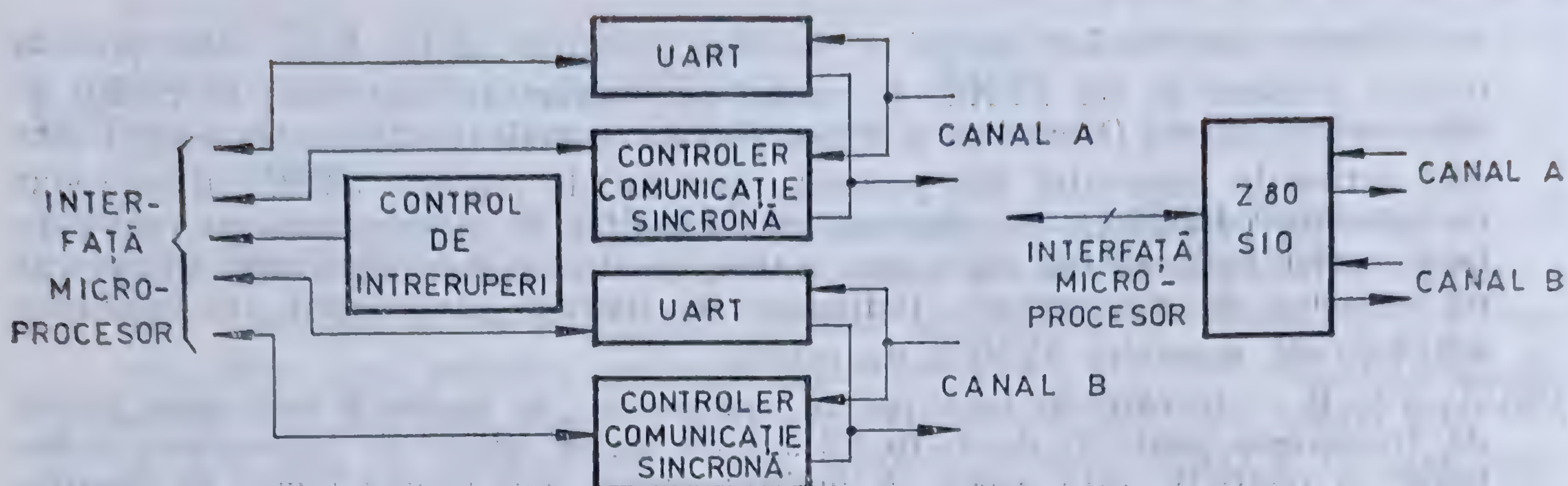


Fig. 5.3 Dispozitivele convenționale înlocuite de circuitul Z80 SIO.

Figura 5.3 ilustrează dispozitivele convenționale pe care le înlocuiește circuitul SIO. În continuare se prezintă posibilitățile de comunicare de date și interacțiunile între unitatea centrală și circuitul SIO.

#### Posibilități de comunicare de date

Circuitul SIO asigură două canale full-duplex, independente, care pot fi programate pentru utilizare în orice protocol comun de comunicare asincronă sau sincronă de date. Figura 5.4 ilustrează câteva dintre cazurile posibile. În continuare se dă o scurtă descriere a lor. Modul HDLC (High Level Data Link Control) este un protocol standard pentru legături de comunicații, stabilit de ISO. Modul SDLC (Synchronous Data Link Control) este un protocol IBM. Ambele sînt orientate pe bit, permițînd comunicări terminal-terminal, terminal-CPU, CPU-CPU etc. În ambele moduri, o stație primară controlează rețeaua și emite comenzi spre stațiile

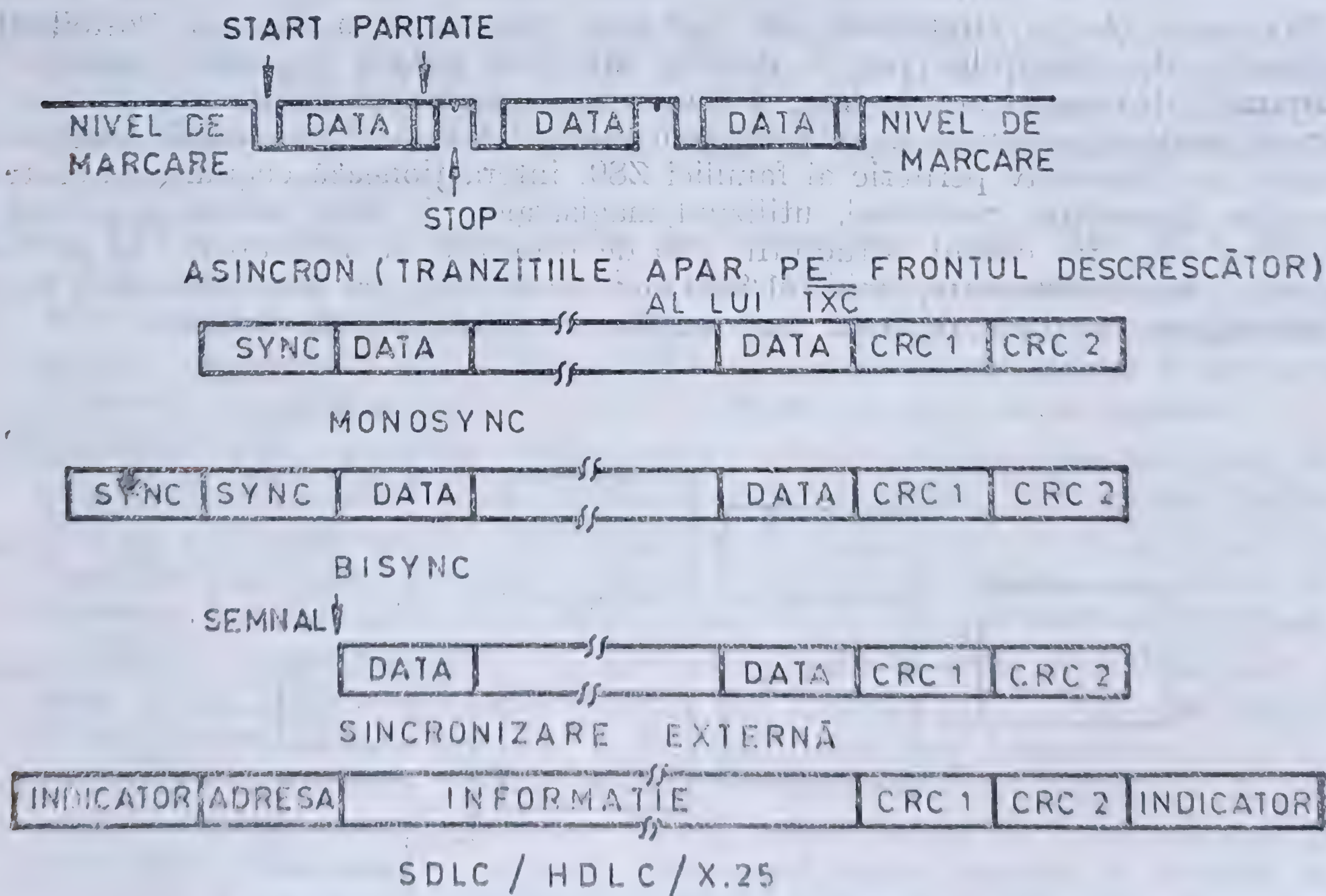


Fig. 5.4 Protocoluri de comunicație acceptate de circuitul Z80 SIO.



secundare, care trimit "răspunsul". Elementul de transmisie este cadrul, ale cărui componente sînt indicatorul de început de 8 biți, un cîmp de adresă, un cîmp de informații și 16 biți de cod CRC, iar în final un indicator de 8 biți (01111110).

### Modul asineron]

Transmisia și recepția pot fi efectuate independent pe fiecare canal, cu 5, 6, 7 sau 8 biți pe caracter și opțional cu paritate pară sau impară. Transmițătorii pot furniza 1,  $1\frac{1}{2}$  sau 2 biți de stop pentru fiecare caracter și pot asigura o ieșire de pauză (Break) în orice moment. Logica de detectare a stării de pauză a receptorului întrerupe unitatea centrală atît la începutul cît și la sfîrșitul unui semnal break, recepționat. Recepția este protejată la paraziți printr-un mecanism care verifică semnalul la un timp corespunzător unei jumătăți de bit după ce se detectează un nivel 0 logic pe intrarea de recepție de date (RXDA sau RXDB). Dacă starea de 0 nu se menține, procesul de asamblare a caracterului nu este declanșat.

Erorile de încadrare sau de viteză depășită (framing errors; overrun errors) sînt detectate și înregistrate împreună cu caracterul (parțial) la care au apărut. Întreruperile cu vector permit servirea rapidă la apariția condițiilor de eroare, utilizînd rutine dedicate. În plus, un proces de verificare înglobat evită interpretarea unei erori de încadrare ca un nou bit de start: eroarea de încadrare duce la adăugarea unui interval corespunzînd unei jumătăți de bit, la momentul în care începe căutarea următorului bit de start.

Circuitul SIO nu necesită semnale de tact de transmisie și recepție simetrice, ceea ce permite utilizarea lui cu circuitul Z80 CTC sau cu alte surse de tact.

Transmițătorul și receptorul pot manevra date cu o viteză de 1, 1/16, 1/32 sau 1/64 din frecvența de tact aplicată pe intrările de tact de transmisie și recepție.

În modurile asincrone, linia  $\overline{\text{SYNC}}$  poate fi programată ca intrare pentru a fi utilizată de exemplu la controlul unui indicator în inel.

### Moduri sincrone

Circuitul SIO acceptă transmisiile sincrone orientate pe octet, ca și pe cele orientate pe bit. În variantele de protocol orientat pe octet, este posibilă sincronizarea cu un caracter de 8 biți (Monosync), cu orice caracter (model) de 16 biți (Bisync) sau cu un semnal extern de sincronizare.

Caracterele conducătoare de sincronizare pot fi înlăturate fără a întrerupe unitatea centrală. Caracterele de sincronizare de 5, 6 sau 7 biți sînt detectate cu șabloane de 8 sau 16 biți în SIO, suprapunînd șablonul mai mare peste caracterele (multiple) de sincronizare care se recepționează, ca în figura 5.5.

Căutarea caracterelor de redondanță ciclică (CRC) pentru modurile sincrone orientate pe octet este întîrziată cu timpul corespunzător unui caracter, astfel încît unitatea centrală poate dezactiva căutarea CRC pentru anumite caractere. Aceasta permite implementarea unor protocoale ca Bisync, al firmei IBM.

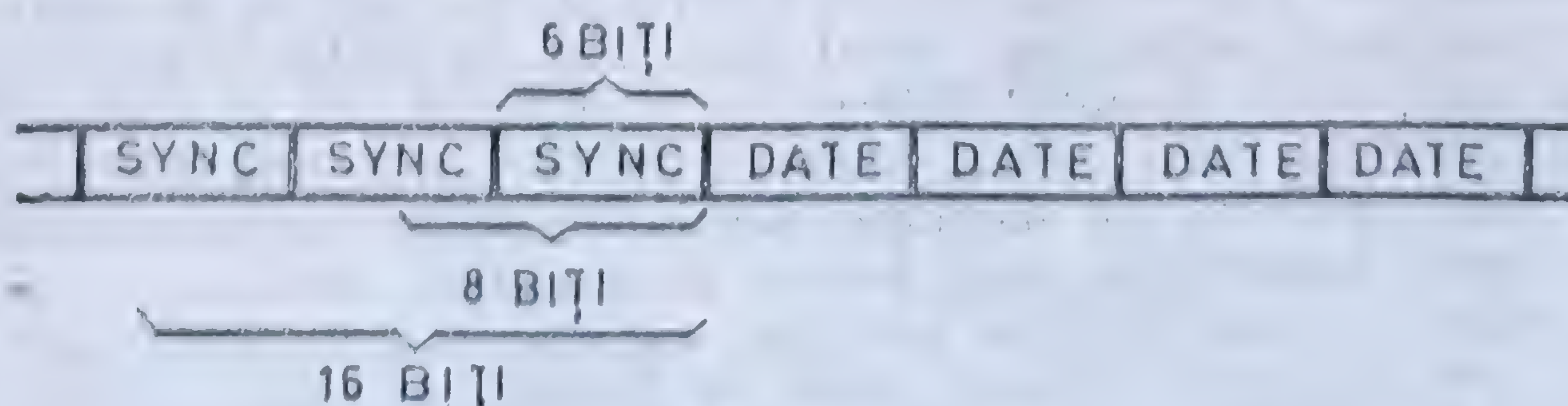


Fig. 5.5 Identificarea caracterelor de sincronizare.



Ambele polinoame de căutare a erorilor, CRC-16 ( $X^{16} + X^{15} + X^2 + 1$ ) și CCITT ( $X^{16} + X^{12} + X^5 + 1$ ) pot fi utilizate. În toate modurile care nu sînt de tip SDLC, generatorul CRC este inițializat cu zerouri. În modurile SDLC, inițializarea se face cu cifre 1. Circuitul SIO poate fi utilizat pentru interfațare cu periferice ca unități de disc flexibil cu sectorizare hard, dar nu poate genera sau verifica coduri CRC pentru discuri compatibile IBM, sectorizate prin soft. Circuitul SIO poate asigura transmiterea automată a datelor CRC cînd nu sînt disponibile pentru transmisie alte date. Aceasta permite transmisia la viteze foarte mari, fără a fi necesară intervenția unității centrale la sfîrșitul mesajului. Cînd nu există date sau caractere de redondanță ciclică (CRC) de transmis în modurile sincrone, transmițătorul inserează caractere de sincronizare de 4 sau 16 biți, indiferent de lungimea programată a caracterelor.

Circuitul SIO acceptă protocoale sincrone, orientate pe bit, ca SDLC și HDLC, realizînd transmiterea automată de indicator, inserare de zerouri și generare de caractere de redondanță ciclică. O comandă specială poate fi utilizată pentru a omite un cadru la transmisie. La sfîrșitul unui mesaj, SIO transmite automat codurile CRC și indicatorul de sfîrșit, cînd tamponul de transmitere devine gol.

Circuitul SIO poate fi utilizat în mod convenabil sub control DMA pentru a asigura transmisia și recepția de mare viteză. La recepție circuitul SIO poate întrerupe unitatea centrală cînd este recepționat primul caracter al unui mesaj. Unitatea centrală validează în continuare circuitul DMA pentru a transfera mesajul în memorie. Circuitul SIO emite în continuare o întrerupere de sfîrșit de cadru și unitatea centrală poate verifica starea mesajului recepționat. Astfel, unitatea centrală este liberă să execute alte sarcini în timpul recepției mesajului. Alte variante de funcționare la recepție sau transmisie vor rezulta în continuare.

### Posibilități ca interfață de intrare/ieșire

Circuitul SIO are ca variante de mod de transfer de date, informații de stare și de control spre sau de la unitatea centrală, modurile de explorare (căutare repetată sau baleiere), întrerupere (cu sau fără vector) și transfer de bloc. Modul de transfer de bloc poate fi implementat și sub control DMA.

În modul de explorare, două registre de stare sînt reactualizate la momente care depind de funcția executată (de exemplu, stare de eroare CRC la sfîrșitul unui mesaj). Cînd unitatea centrală funcționează într-un mod de explorare, unul dintre cele două registre de stare din SIO, RR0 sau RR1 este utilizat pentru a indica situația în care circuitul SIO are date disponibile sau în care are nevoie de date.

Unitatea centrală examinează starea registrului RR0. Biții D0 și D2 arată că este necesară recepția sau transmisia de date. Sînt indicate de asemenea erori sau alte condiții speciale de stare. Starea condițiilor speciale de recepție din RR1 se citește doar după recepționarea unui caracter. În funcție de conținutul acestui registru, unitatea centrală va scrie date, va citi date sau își va continua activitatea.

Toate modurile de întrerupere sînt dezactivate cînd se lucrează cu dispozitivul în modul de explorare.

În modul de întrerupere, pentru care circuitul SIO dispune de o logică pentru a asigura servirea rapidă a întreruperilor în aplicații în timp real, vectorul de întrerupere este conținut într-un registru de control și un registru de stare din canalul B. Cînd este programat astfel, prin bitul STAREA AFECTEAZA VECTORUL, din registrul de scriere WR1, circuitul SIO poate modifica 3 biți ai vectorului de întrerupere din registrul de stare, astfel încît să indice direct una dintre cele 8 rutine de servire a întreruperilor, din memorie, servind condiții din ambele canale și cele mai multe dintre necesitățile unei rutine de analiză a stării.

Întreruperile la transmisie, la recepție, externe sau de stare sînt principalele variante care pot să apară. Fiecare sursă de întrerupere este validată sub control de program, cu canalul A avînd o prioritate mai mare decît canalul B și cu între-



ruperile la recepție, transmisie, externe sau de stare în această ordine de prioritate pentru fiecare canal. Când sînt validate întreruperile la transmisie, unitatea centrală este întreruptă de faptul că registrul tampon de transmisie a devenit gol (ceea ce implică faptul că transmițătorul trebuie să fi avut un caracter — dată înscris în el înainte de a fi devenit gol). Receptorul poate întrerupe unitatea centrală într-unul dintre cele 2 moduri posibile: întrerupere la primul caracter recepționat sau întrerupere la toate caracterele recepționate.

Întreruperea pe primul caracter recepționat este utilizată în mod obișnuit în modul de transfer de bloc. Întreruperea pe toate caracterele recepționate are opțiunea de a modifica vectorul de întrerupere în cazul apariției unei erori de paritate. În ambele moduri de întrerupere de mai sus se va genera o întrerupere în condiții speciale la recepția unui caracter sau a unui mesaj (de exemplu, întrerupere la sfîrșit de cadru, în SDLC). Aceasta înseamnă că se va genera o întrerupere pe condiția specială de recepție, numai dacă este selectat modul de întreruperi pe primul caracter recepționat sau pe toate caracterele recepționate.

La întreruperea pe primul caracter, o întrerupere poate să apară din condiții speciale de recepție (cu excepția erorii de paritate), după întreruperea pe primul caracter recepționat (de exemplu, întrerupere la depășire de viteză la recepție).

Principala funcție a întreruperilor externe sau de stare este de a controla tranzițiile semnalelor CTS (gata pentru transmisie), DCD (detectarea purtătoarei de date) și SYNC (sincronizare). În plus, întreruperile externe/de stare sînt cauzate de o condiție CRC la transmisie sau de detectarea unei secvențe de pauză (în modul asincron) sau a unei secvențe de evitare (abort, în modul SDLC) în cadrul șirului de date. Întreruperea cauzată de secvența de pauză sau de evitare permite circuitului SIO să întrerupă atunci cînd secvența este detectată sau cînd este terminată. Aceasta facilitează terminarea corectă a mesajului curent, inițializarea corectă a mesajului următor și încadrarea exactă în timp a condiției de pauză/evitare în logica externă.

Într-un sistem Z80 tipic (figura 5.6) direcționarea cu vectori a întreruperilor de la circuitul SIO este automată: circuitul SIO își trece vectorul de întreruperi de 8 biți, modificabil intern, spre unitatea centrală care adaugă cei 8 biți din registrul de întreruperi I pentru a forma adresa de memorie a tabelului rutinelor de întrerupere. Acest tabel conține adresa de început a rutinei de întrerupere. Se asigură astfel un transfer indirect al controlului unității centrale la rutina de întrerupere, astfel încît următoarea instrucțiune executată după o accep-

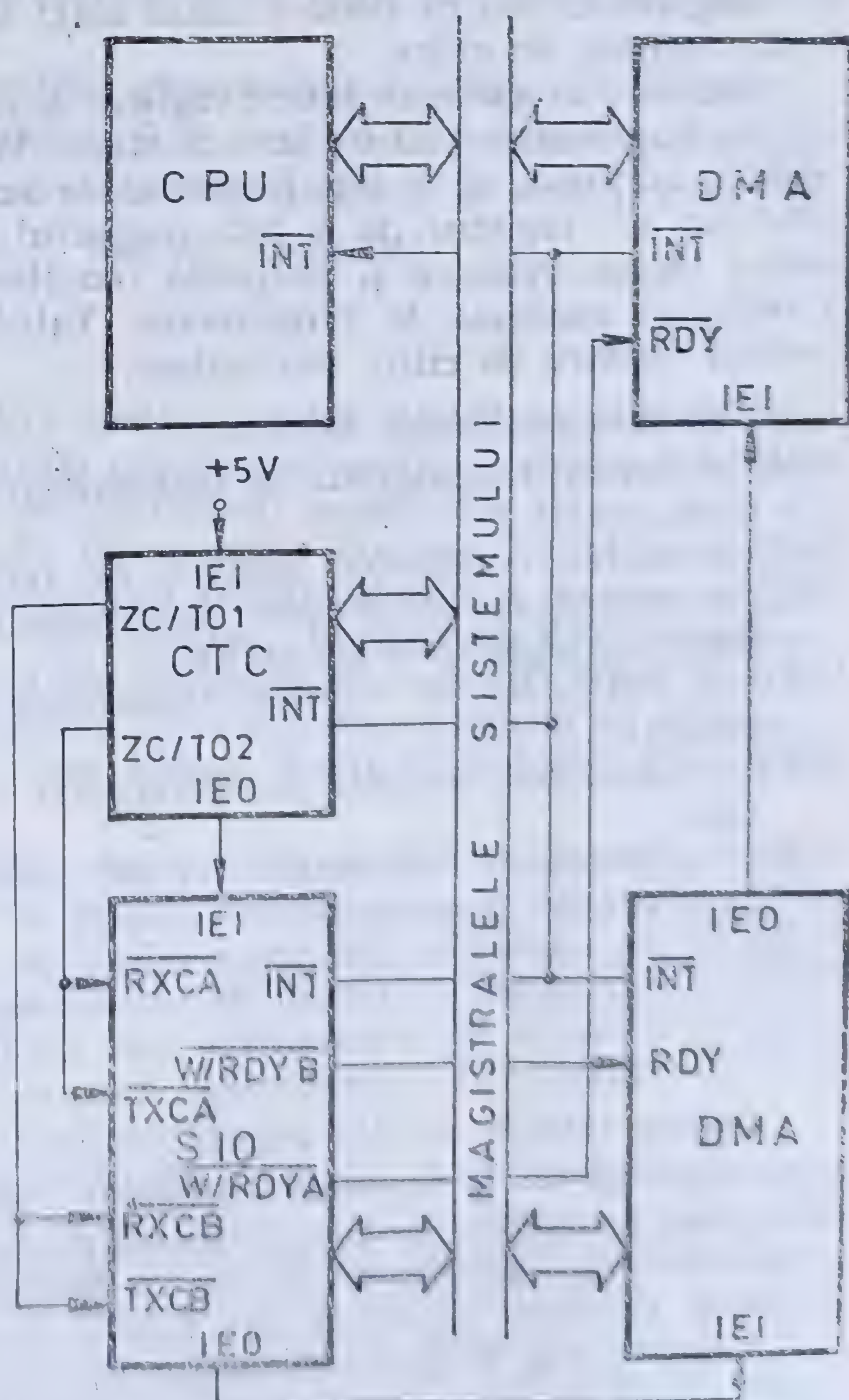


Fig. 5.6 Sistem Z80 tipic.



tare de întrerupere de către unitatea centrală, este prima instrucțiune a rutinei de întrerupere.

În modul de transfer de bloc CPU/DMA, circuitul SIO acceptă atât transferul de bloc cu unitatea centrală, cât și cu circuite de control DMA (Z80 DMA, sau alte variante). Modul de transfer de bloc utilizează semnalul de ieșire Wait/Ready (Așteaptă/Gata), care este selectat cu 3 biți într-un registru de control intern (D7, D6, D5 din registrul WR1). Semnalul de ieșire Wait/Ready poate fi programat ca ieșire de așteptare,  $\overline{\text{WAIT}}$ , în modul de transfer de bloc cu unitatea centrală, sau ca linie  $\overline{\text{READY}}$ , în modul de transfer de bloc DMA. Semnalul  $\overline{\text{READY}}$  al circuitului SIO arată circuitului de control DMA că SIO este gata să transfere date spre, sau de la memorie. Ieșirea  $\overline{\text{WAIT}}$  arată unității centrale că SIO nu este gata de transfere date, cerind astfel unității centrale să extindă ciclul de intrare/ieșire.

### Structura internă a circuitului Z80 SIO

Structura internă a dispozitivului include o interfață cu unitatea centrală Z80 CPU, logică internă de control și de întreruperi și 2 canale de comunicație full-duplex. Fiecare canal conține setul său propriu de registre de control și stare (pentru scriere și pentru citire), și logică de control și de stare care asigură interfațarea cu modemuri sau alte dispozitive externe.

Registrele pentru fiecare canal sînt: WRO—WR7, registre de scriere și RR0—RR2, registre de citire.

Grupul de registre include 5 registre de control de 8 biți, 2 registre pentru caractere de sincronizare și 2 registre de stare. Vectorul de întreruperi este înscris într-un registru adițional de 8 biți (registrul de scriere 2), în canalul B, care poate fi citit printr-un alt registru de 8 biți (registrul de citire 2) din canalul B. Rolul biților pentru fiecare registru și gruparea funcțională este configurată pentru a simplifica și organiza procesul de programare. Tabelul de mai jos listează funcțiile asociate fiecărui registru de citire sau scriere.

#### Funcțiile registrelor de citire

RR0 — starea tamponului de transmisie/recepție, starea întreruperilor și starea externă

RR1 — starea condițiilor speciale de recepție

RR2 — vector de întrerupere modificat (canalul B)

#### Funcțiile registrelor de scriere

WR0 — indicatori de registre, inițializare CRC, comenzi de inițializare în diferite moduri

WR1 — definirea modului de întreruperi la transmisie/recepție și de transmisie de date

WR2 — vector de întreruperi (numai pentru canalul B)

WR3 — control și parametri de recepție

WR4 — parametri și diferite moduri de transmisie/recepție

WR5 — parametri și control de transmisie

WR6 — caracter de sincronizare sau cîmp de adresă SDLC

WR7 — caracter de sincronizare sau indicator SDLC

Logica ambelor canale asigură formatul, sincronizarea și validarea datelor transferate spre și de la interfața canalului. Intrările de control pentru modem,  $\overline{\text{CTS}}$  și  $\overline{\text{DCD}}$ , sînt controlate de logica externă de control și de stare, sub supravegherea programului. Toate semnalele externe de control și de stare a logicii sînt de uz general și pot fi utilizate și pentru alte funcții decît controlul modemului.

Calea datelor la transmisie și recepție este identică pentru ambele canale. Receptorul are 3 registre tampon de 8 biți, aranjate în ordine FIFO (primul intrat—primul ieșit), și un registru de deplasare la recepție, de 8 biți. În modul asincron,



datele intră în cele 3 registre FIFO dacă au 7 sau 8 biți/caracter, sau în registrul de deplasare de 8 biți dacă au 5 sau 6 biți/caracter. Această schemă creează timp adițional pentru ca unitatea centrală să poată servi o întrerupere la începutul unui bloc de date la viteză mare. Datele de intrare sînt ghidate pe una dintre căile de date sau CRC posibile, în funcție de modul selectat și, în modurile asincrone, de lungimea caracterului.

Transmițătorul are un registru tampon de 8 biți pentru date, încărcat de pe magistrala internă de date, și un registru de deplasare la transmisie de 20 de biți, care poate fi încărcat de la tamponurile caracterelor de sincronizare sau de la registrul de transmisie a datelor. În funcție de modul de operare, datele de ieșire sînt ghidate pe una dintre cele 4 căi principale existente, înainte de a fi transmise pe ieșirea de transmitere a datelor, TXD.

### Programarea circuitului Z80 SIO

Programul sistemului emite la început o serie de comenzi care inițializează modul de funcționare de bază și, în continuare, alte comenzi care precizează condiții în cadrul modului selectat. De exemplu, modul asincron, lungimea caracterului, frecvența de tact, numărul de biți de stop, paritatea pară sau impară, trebuie fixate la început, iar în continuare modul de întreruperi și, în final, validarea receptorului sau transmițătorului.

Ambele canale conțin registre care trebuie programate de către sistem înaintea funcționării. Intrarea de selecție a canalului,  $B/\bar{A}$  și intrarea de control sau date,  $C/\bar{D}$ , sînt intrările de control ale adresei structurii de comandă și sînt în mod obișnuit controlate de magistrala de adrese a unității centrale.

### Registrele de citire

Circuitul SIO conține 3 registre de citire pentru canalul B și 2 registre de citire pentru canalul A (RR0—RR2 din fig. 5.7), care pot fi citite pentru a obține informații de stare. Registrul RR2 conține vectorul de întreruperi modificabil intern și se află numai în setul de registre al canalului B. Informațiile de stare includ condiții de eroare, un vector de întrerupere și semnale standard pentru interfețele de comunicație. Pentru a citi conținutul unui registru de citire selectat, în afară de RR0, programul sistemului trebuie să înscrie la început octetul indicator în registrul WR0, în mod identic cu o operație de scriere a unui registru. În continuare, executînd o instrucțiune de citire, conținutul registrului de citire adresat poate fi citit de către unitatea centrală.

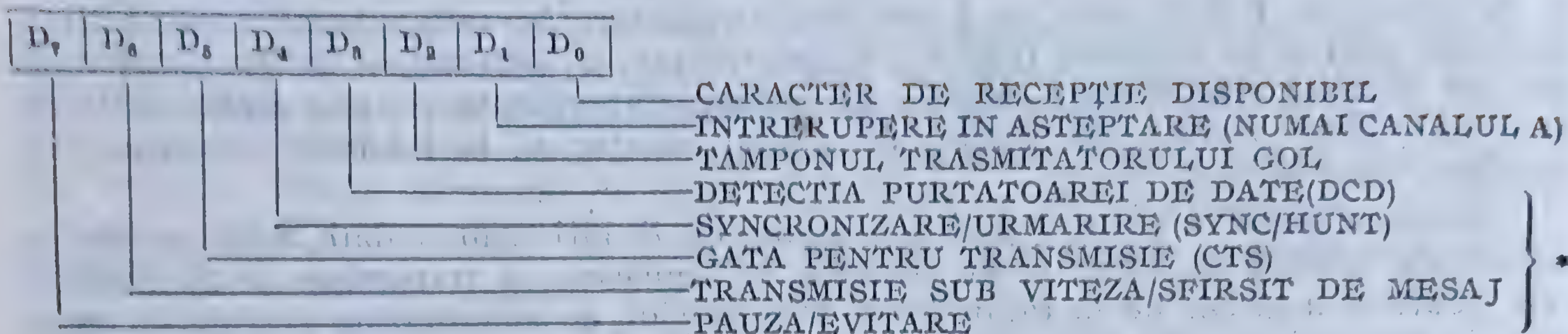
Biții de stare ai registrelor RR0 și RR1 sînt grupați pentru a simplifica operația de control a stării. De exemplu, cînd vectorul de întreruperi indică faptul că a apărut o întrerupere sau o condiție specială la recepție, toți biții de eroare vizați pot fi citați dintr-un singur registru (RR1).

### Registrele de scriere

Circuitul SIO conține 8 registre de scriere pentru canalul B și 7 registre pentru canalul A (WR0—WR7 din fig. 5.8), care sînt programate separat pentru a da caracteristicile funcționale ale canalului. Registrul WR2 conține vectorul de întreruperi pentru ambele canale și se află numai în setul de registre al canalului B. Cu excepția lui WR0, programarea registrelor de scriere necesită 2 octeți. Primul octet se adresează lui WR0 și conține 3 biți, D0 ÷ D2, care indică registru selectat. Al doilea octet este adevăratul cuvînt de control înscris în registru pentru a stabili anumite caracteristici funcționale. Registrul WR0 constituie un caz special, prin aceea că

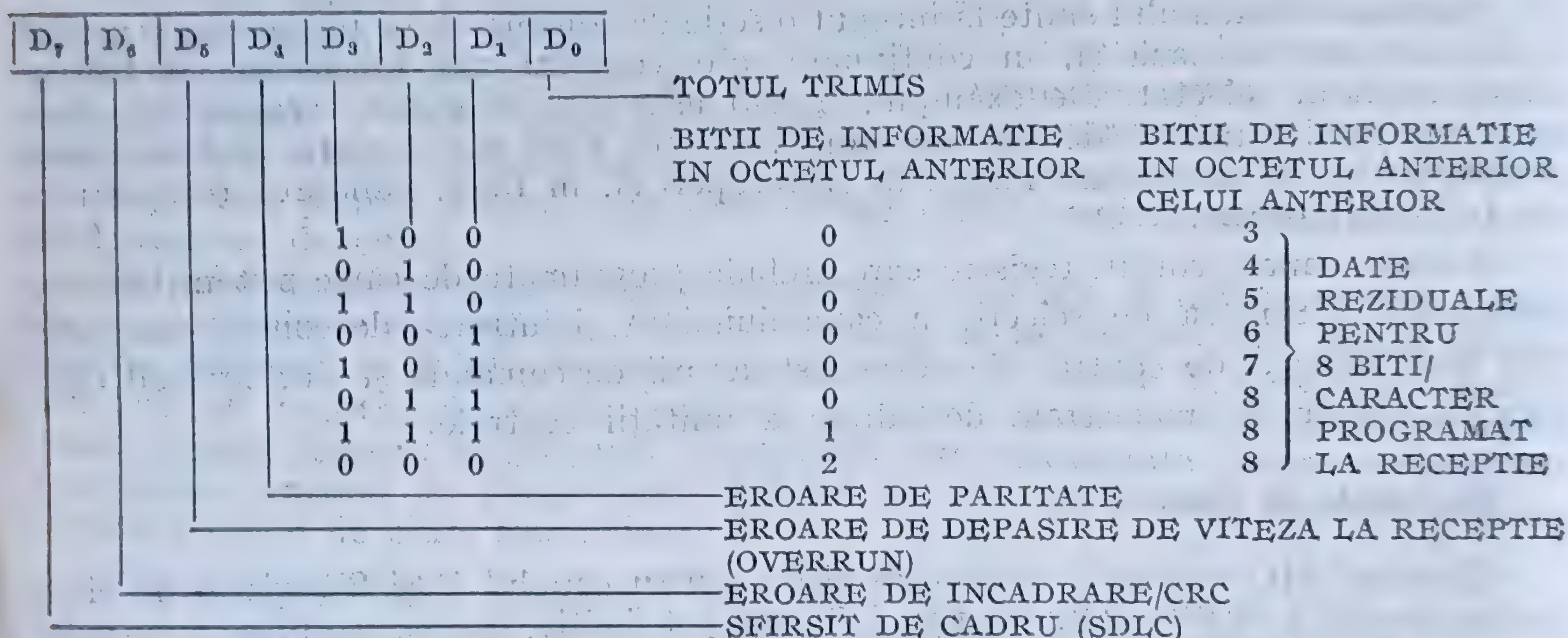


### REGISTRUL DE CITIRE 0, RR0



\* UTILIZATE IN MODUL DE INTRERUPERE DE STARE/EXTERNĂ

### REGISTRUL DE CITIRE 1, RR1\*\*



\*\* UTILIZAT IN MODUL DE CONDITII SPECIALE LA RECEPTIE

### REGISTRUL DE CITIRE 2, RR2

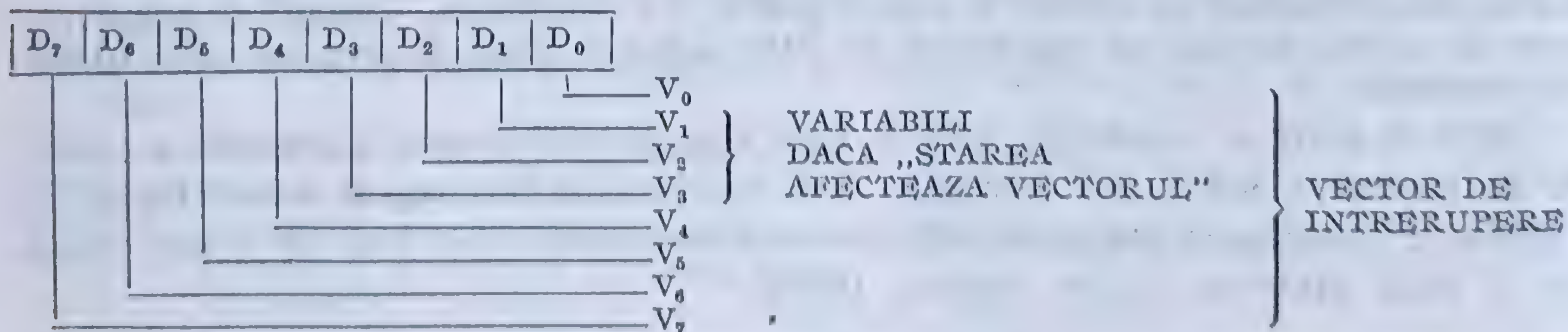


Fig. 5.7 Funcțiile biților din registrele de citire.

toate comenzile de bază pot fi înscrise în el cu un singur octet. Inițializarea (internă sau externă) aduce biții D0—D2 la starea în care indică pe WR0. Aceasta implică faptul că inițializarea unui canal nu trebuie combinată cu indicarea nici unui alt registru. Semnificația biților registrelor de citire și de scriere este descrisă în continuare.



# REGISTRUL DE SCRIERE 0, WR0

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

					PTR2	PTR1	PTR0	BITI INDICATORI
					0	0	0	REGISTRUL 0
					0	0	1	REGISTRUL 1
					0	1	0	REGISTRUL 2
					0	1	1	REGISTRUL 3
					1	0	0	REGISTRUL 4
					1	0	1	REGISTRUL 5
					1	1	0	REGISTRUL 6
					1	1	1	REGISTRUL 7
		CMD2	CMD1	CMD0	1	1	1	
		0	0	0				COD NUL
		0	0	1				OMITE TRIMITAREA (SDLC)
		0	1	0				INITIALIZARE INTRERUPERI EXTERNE/DE STARE
		0	1	1				INITIALIZARE DE CANAL
		1	0	0				VALIDEAZA INTRERUPERILE PE URMATORUL CARACTER RECEPTIONAT
		1	0	1				INITIALIZAREINTRE RUPERE LA TRANSMISIE IN ASTEPTARE
COD RESET	1		1	0				INITIALIZARE ERORI
CRC1 CRC0	1		1	1				REVENIRE DIN INTRERUPERE (NUMAI PTR. CANALUL A)
0	0							COD NUL
0	1							INITIALIZEAZA VERIFICAREA CRC LA RECEPTIE
1	0							INITIALIZEAZA GENERATORUL CRC LA TRANSMISIE
1	1							INITIALIZEAZA BISTABILUL DE TRANSMISIE SUB VITEZA/SFIRSIT MESAJ

## REGISTRUL DE SCRIERE 1, WR1

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

								VALIDAREA INTRERUPERILOR DE STARE/EXTERNE
								VALIDAREA INTRERUPERILOR LA TRANSMITERE
								STAREA AFECTEAZA VECTORUL (NUMAI PT. CANAL B)
					0	0		DEZACTIVAREA INTRERUPERILOR LA RECEPTIE
					0	1		INTRERUPERE LA RECEPTIA PRIMULUI CARACTER
					1	0		INTRERUPERE LA RECEPTIA TUTUROR CARACTERELOR (PARITATEA AFECTEAZA VECTORUL)
					1	1		INTRERUPERE LA RECEPTIA TUTUROR CARACTERELOR (PARITATEA NU AFECTEAZA VECTORUL)
								WAIT/READY LA RECEPTIE/TRANSMISIE
								FUNCTIA WAIT/READY
								VALIDEAZA WAIT/READY

\* SAU PE CONDIȚII SPECIALE

## REGISTRUL DE SCRIERE 2, WR2 (NUMAI PENTRU CANALUL B)

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

								V <sub>0</sub>
								V <sub>1</sub>
								V <sub>2</sub>
								V <sub>3</sub>
								V <sub>4</sub>
								V <sub>5</sub>
								V <sub>6</sub>
								V <sub>7</sub>

VECTOR DE INTRERUPERE

Fig. 5.8 partea I.







# REGISTRUL DE SCRIERE 7, WR7

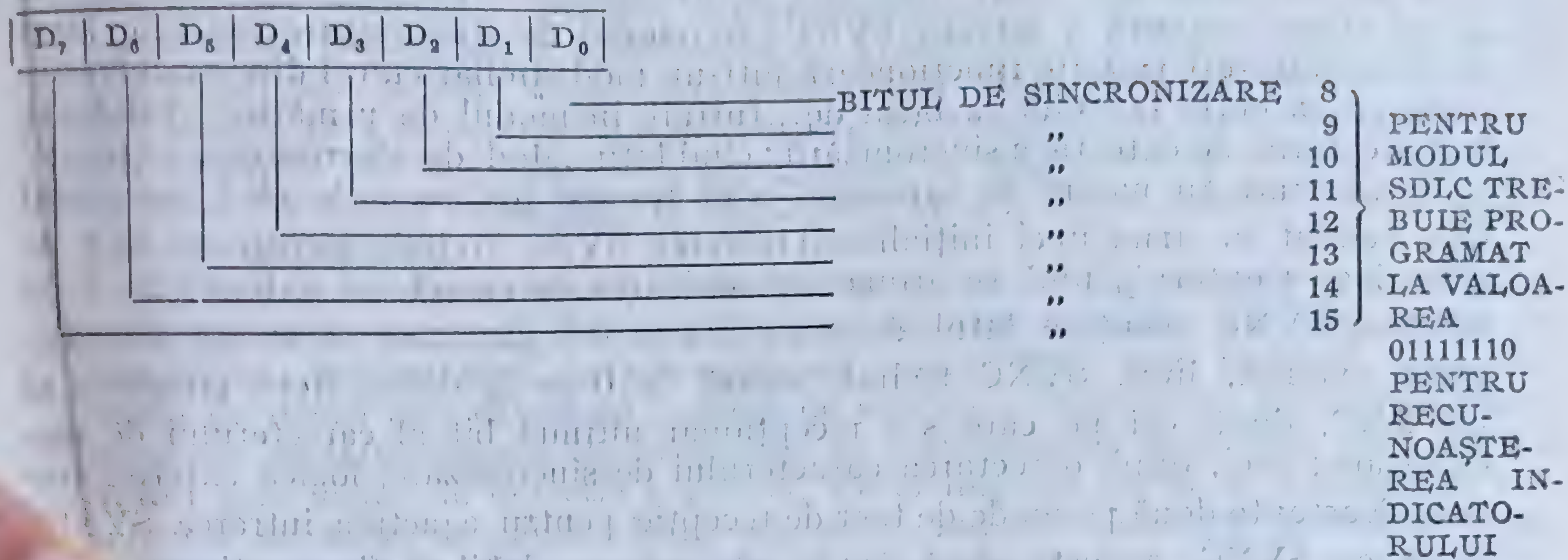


Fig. 5.8. partea III (continuare)

Fig. 5.8 Funcțiile biților din registrele de scriere.

Registrul RR0 — conține starea registrelor tampon de recepție și transmisie, intrările DCD, CTS și SYNC, bistabilul de „Transmisie sub viteză/Sfârșit de mesaj” și cel de „Pauză/Evitare”; semnificația fiecărui bit este descrisă în continuare:

- D0 — caracter de recepție disponibil; este înscris când există cel puțin un caracter disponibil în registrele tampon de recepție; este șters ( $D0 = 0$ ) când tamponul FIFO este complet gol.
- D1 — întrerupere în așteptare; este înscris de orice condiție de întrerupere care apare la SIO; bitul se poate citi însă numai în canalul A și este utilizat în principal în aplicații în care nu există disponibil un vector de întrerupere; în timpul rutinei de servire a întreruperii în aceste aplicații, acest bit indică existența condițiilor de întrerupere în circuitul SIO, ceea ce elimină necesitatea analizării biților din registrele RR0 ale canalelor A și B; bitul D1 este șters când s-au satisfăcut toate condițiile de întrerupere; în canalul B acest bit este întotdeauna 0.
- D2 — tamponul transmițătorului gol; este înscris ori de câte ori registrul tampon al transmițătorului devine gol, cu excepția cazului în care se transmite un caracter CRC într-un mod SDLC sau sincron; bitul este șters când se încarcă un caracter în tamponul transmițătorului; este înscris după inițializare (RESET).
- D3 — detecția purtătoarei de date; arată starea inversată a intrării DCD la momentul ultimei schimbări a oricăruia dintre cei 5 biți de stare/externi (DCD, CTS, Sincronizare/Urmărire, Pauză/Evitare sau Transmisie sub viteză/Sfârșit de mesaj); orice tranziție a intrării DCD determină captarea în bitul DCD și apare o întrerupere externă/de stare; pentru a citi starea curentă a bitului DCD, trebuie citit imediat după o comandă de inițializare a întreruperilor de stare/externe.
- D4 — sincronizare/urmărire; este controlat în mod diferit în modurile asincrone, sincrone sau SDLC; în modurile asincrone, funcționarea este similară cu a bitului DCD, cu deosebirea că D4 indică starea intrării SYNC; orice tranziție de la 1 la 0 pe linia SYNC înscrie acest bit și determină o întrerupere de stare/externă (dacă este validată); comanda de inițializare a întreruperilor de stare/externe este emisă pentru a șterge întreruperea; o tranziție de la 0 la 1 șterge acest bit și fixează o întrerupere de stare/externă; când întreruperea apare ca urmare a schimbării de stare a oricărei alte intrări sau condiții, acest bit indică



starea inversată a liniei  $\overline{\text{SYNC}}$  în momentul schimbării; bitul trebuie citit imediat după o comandă de inițializare a întreruperilor externe/de stare, pentru a citi starea curentă a intrării  $\overline{\text{SYNC}}$ ; în modul de sincronizare externă, bitul de Sincronizare/Urmărire funcționează într-un mod similar cu cel din modul asincron, cu excepția faptului că bitul de „Intrare în modul de urmărire” validează logica externă de detecție a sincronizării; când biții „Mod de sincronizare externă” și de „Intrare în modul de urmărire” sînt înscriși (de exemplu cînd receptorul este validat în urma unei inițializări) intrarea  $\overline{\text{SYNC}}$  trebuie menținută la 1 de către logica externă, pînă se atinge sincronizarea cu caracterul extern; un 1 pe intrarea  $\overline{\text{SYNC}}$  menține bitul de stare D4 la 0 logic; cînd se atinge sincronizarea externă, linia  $\overline{\text{SYNC}}$  trebuie adusă la 0 pe la doilea front crescător al lui  $\overline{\text{RXC}}$ , după cel pe care s-a recepționat ultimul bit al caracterului de sincronizare; deci, după detectarea caracterului de sincronizare, logica externă trebuie să aștepte două perioade de tact de recepție pentru a activa intrarea  $\overline{\text{SYNC}}$ ; după ce  $\overline{\text{SYNC}}$  a fost adusă la 0, este recomandabil să fie menținută astfel pînă cînd unitatea centrală informează logica externă de sincronizare că s-a pierdut sincronizarea sau că începe un nou mesaj; tranziția de la 1 la 0 pe intrarea  $\overline{\text{SYNC}}$  înscrie cu 1 bitul de „Sincronizare/Urmărire” (Sync/Hunt), care, în continuare determină întreruperea de stare/externă. Unitatea centrală trebuie să șteargă întreruperea emițînd comanda de inițializare a întreruperilor de stare/externe; cînd intrarea  $\overline{\text{SYNC}}$  trece din nou la 1, se generează o altă întrerupere de stare/externă, care trebuie de asemenea să fie ștearsă; bitul de control al modului de intrare în urmărire este înscris ori de cîte ori se pierde sincronizarea caracterelor sau este detectat sfîrșitul mesajului; în acest caz, circuitul Z80 SIO caută din nou o tranziție de la 1 la 0 a intrării  $\overline{\text{SYNC}}$  și funcționarea se repetă după algoritmul de mai sus; aceasta implică faptul că unitatea centrală trebuie să informeze logica externă că s-a pierdut sincronizarea caracterelor și că Z80 SIO așteaptă ca semnalul  $\overline{\text{SYNC}}$  să devină activ; în modurile de recepție MONOSYNC și BISYNC, bitul de stare Sync/Hunt este inițial înscris cu 1, de bitul de Mod de intrare în urmărire; bitul de Sincronizare/Urmărire (Sync/Hunt) este șters cînd Z80 SIO stabilește sincronizarea caracterelor; tranziția de la 1 la 0 a bitului Sync/Hunt determină o întrerupere externă/de stare care trebuie să fie ștearsă de unitatea centrală prin emiterea comenzii de Inițializare a întreruperilor de stare/externe; aceasta permite circuitului Z80 SIO să detecteze următoarea tranziție a altui bit de stare/extern; cînd unitatea centrală detectează sfîrșitul mesajului sau cînd s-a pierdut sincronizarea, se înscrie bitul de Intrare în modul de urmărire, care, în continuare aduce bitul Sync/Hunt la 1; tranziția de la 0 la 1 a bitului Sync/Hunt determină o întrerupere de stare/externă, care trebuie de asemenea să fie ștearsă prin comanda de Inițializare a întreruperilor de stare/externe; de notat faptul că linia  $\overline{\text{SYNC}}$  funcționează ca ieșire în acest mod, și trece la 0 de fiecare dată cînd este detectat un caracter de sincronizare în șirul de date; în modul SDLC, bitul de Sincronizare/Urmărire este inițial înscris de bitul de Intrare în modul de urmărire, sau cînd receptorul este dezactivat; în oricare caz, este șters cînd Z80 SIO detectează indicatorul de început al primului cadru; se generează și o întrerupere de stare/externă, care trebuie tratată ca mai sus; spre deosebire de modurile Monosync și Bisync, după ce s-a șters bitul Sync/Hunt în modul SDLC, el nu trebuie înscris cînd se detectează sfîrșitul mesajului; circuitul Z80 SIO menține automat sincronizarea; singurul mod de a înscrie din nou bitul Sync/Hunt este cu ajutorul bitului de Intrare în modul de urmărire, sau dezactivînd receptorul.

D5 — Gata pentru transmisie, CTS; este similar cu bitul DCD, dar indică starea inversată a liniei  $\overline{\text{CTS}}$ .



D6 — transmisie sub viteză/sfârșit de mesaj; este înscris în urma unei inițializări (internă sau externă); singura comandă care poate să șteargă acest bit este comanda de inițializare a bistabilului de Transmisie sub viteză/Sfârșit de mesaj (D6 și D7 din WR0); când apare condiția de Transmisie sub viteză, bitul D6 este înscris, ceea ce determină o întrerupere de stare/externă, care trebuie ștearsă din comanda de inițializare a întreruperilor de stare/externe (WR0); acest bit de stare are un rol important, în combinație cu alți biți de control pentru operațiile de transmitere.

D7 — pauză/evitare (break/abort); este înscris în modul de recepție asincron când se detectează o secvență de pauză în șirul de date (caracter nul plus eroare de încadrare); întreruperea de stare/externă apare, dacă este validată, când secvența de pauză este detectată; rutina de servire a întreruperii trebuie să emită comanda de inițializare a întreruperilor de stare/externe (prin biții D3—D5 din WR0) spre logica de detecție a pauzei, astfel încât să poată fi recunoscută terminarea secvenței de pauză; bitul este șters când este detectată terminarea secvenței de pauză în șirul de date de intrare; aceasta determină apariția întreruperii de stare/externă; comanda de inițializare a întreruperilor de stare/externe trebuie să fie emisă pentru a permite logicii de detecție a pauzei să caute următoarea secvență de pauză; un singur caracter nul, fără legătură cu celelalte caractere, va fi prezent în receptor după terminarea pauzei; acest caracter ar trebui citit și înlăturat; în modul de recepție SDLC, acest bit de stare este înscris de detecția unei secvențe de evitare (7 sau mai multe cifre „1”); întreruperea de stare/externă este tratată ca și în cazul descris pentru pauză; bitul D7 nu este utilizat în modul de recepție sincronă.

Registrul RR1 — conține biții de stare pentru condiții speciale la recepție și codurile reziduale în câmpul I (de informații) în modul de recepție SDLC. Semnificația fiecărui bit este:

D0 — totul transmis; este înscris în modurile asincrone atunci când toate caracterele din transmitător au fost transmise; tranzițiile acestui bit nu determină întreruperi; bitul este întotdeauna înscris în modurile sincrone.

D1—D3 — date reziduale; au rol în modul SDLC; în cazul modului de recepție SDLC în care câmpul de informații nu este un multiplu întreg al lungimii caracterului, acești 3 biți indică lungimea câmpului de informații; codurile D1 ÷ D3 au sens numai pentru transferuri în care bitul „Sfârșit de cadru” este înscris (SDLC); pentru o lungime de caracter de recepție de 8 biți/caracter, semnificația codurilor reziduale D1 ÷ D3 este dată în fig. 5.7; dacă lungimea caracterului utilizat în câmpul de informații diferă de 8 biți, se poate construi un tabel similar celui pentru 8 biți, pentru fiecare lungime diferită de caracter; dacă nu există date reziduale, deci dacă marginea ultimului caracter coincide cu marginea câmpului de informații și CRC, codurile reziduale vor fi:

Biți/caracter	D3	D2	D1
8	0	1	1
7	0	0	0
6	0	1	0
5	0	0	1

D4 — eroare de paritate; este înscris în cazul în care paritatea este validată, pentru acele caractere a căror paritate nu coincide cu cea programată (pară sau impară); bitul este captat, deci dacă apare o eroare, rămâne înscris pînă când se dă comanda de inițializare a erorilor (prin WR0).

D5 — eroare de depășire de viteză la recepție; arată că au fost recepționate mai mult de 3 caractere fără a fi citite de unitatea centrală; acest bit este înscris numai pentru caracterul care a fost suprapus, dar la citirea acestui caracter, D5 rămâne 1 pînă este șters prin comanda de inițializare a erorilor; dacă este



validată afectarea vectorului de întrerupere prin bitul „Starea afectează vectorul”, caracterul care a fost supraînscris determină o întrerupere cu un vector de „Condiții speciale la recepție”.

D6 — eroare de încadrare/CRC; este înscris, în modurile asincrone dacă apare o eroare de încadrare, pentru caracterul de recepție la care a apărut această eroare; bitul nu este captat, deci se schimbă în funcție de apariția erorii de încadrare la fiecare caracter; detecția unei erori de încadrare adaugă un timp adițional de o jumătate de bit la timpul afectat caracterului, astfel încât eroarea de încadrare să nu fie interpretată ca un nou bit de start; în modurile sincrone și SDLC, acest bit indică rezultatul comparării codului CRC calculat de vericator cu cel recepționat; bitul este șters emitând o comandă de inițializare a erorilor; bitul nu este captat, deci se reactualizează la fiecare caracter; în modurile sincrone, utilizat pentru stare și eroare CRC, bitul este în mod uzual înscris, cele mai multe combinații de bit dând un cod CRC nenul, cu excepția mesajelor terminate corect.

D7 — sfârșit de cadru; se utilizează numai în modul SDLC și arată că a fost primit un indicator corect de terminare și că sînt de asemenea avalide și codurile reziduale și de eroare CRC; bitul poate fi șters emitând comanda de inițializare a erorilor; este de asemenea șters de primul caracter al următorului cadru.

Registrul RR2 — conține vectorul de întreruperi înscris în WR2, dacă bitul „Starea afectează vectorul” nu este înscris; dacă acest bit este înscris, RR2 conține vectorul modificat, așa cum s-a arătat la descrierea bitului D2 din registrul de scriere WR1; cînd se citește acest vector, rezultatul citirii este determinat de condiția de întrerupere cu cea mai mare prioritate care există în momentul citirii; dacă nu există întreruperi în așteptare, vectorul este modificat cu  $V3=0$ ,  $V2=1$  și  $V1=1$ ; acest registru poate fi citit numai în canalul B.

Registrul WR0 — indică oricare alt registru, stabilește comanda de bază și alți parametri.

D0 — D2 — biții indicatori; arată din care registru se va citi sau în care se va înscrie octetul următor; primul octet în fiecare canal, după o inițializare (prin comandă sau din exterior) se adresează registrului WR0; oricărei citiri sau scrieri adresate unui alt registru decît WR0 trebuie să-i urmeze una adresată lui WR0.

D3 — D5 — codifică cele 7 comenzi de bază ale circuitului Z80 SIO:

- comanda 0 — codul nul; nu are efect; SIO nu execută nimic în timp ce biții indicatori sînt fixați pentru următorul octet.

- comanda 1 — evită trimiterea (send abort); se utilizează în modul SDLC pentru a genera o secvență de 8 pînă la 13 biți „1”.

- comanda 2 — inițializare întreruperi de stare/externe; activează din nou biții de stare din RR0, captați la apariția unei întreruperi de stare/externă (de exemplu, o schimbare pe o linie de modem sau o condiție de pauză) și permite apariția unor noi întreruperi; captarea biților de stare din RR0 „memorează” impulsurile scurte pînă cînd unitatea centrală are timp să citească schimbarea.

- comanda 3 — inițializare de canal; are același rol cu comanda de inițializare externă (reset extern), dar numai pe un singur canal; inițializarea canalului A inițializează și logica de prioritate a întreruperilor; toate registrele de control ale canalelor trebuie rescrise după o comandă de inițializare; după inițializarea canalului sînt necesare 4 perioade de tact înainte de a înscrie alte comenzi (este suficient timpul necesar aducerii codului următor de operație din memorie).

- comanda 4 — validează întreruperile pe următorul caracter recepționat; reactivează modul de întrerupere pe primul caracter recepționat după primirea fiecărui mesaj complet, pentru a pregăti circuitul SIO pentru următorul mesaj.



- comanda 5 — inițializarea întreruperii la transmisie în așteptare; are rol în cazurile în care nu mai sînt caractere de transmis (de exemplu la sfîrșitul unui mesaj), în modul de întreruperi validate la transmisie, cînd transmițătorul întrerupe dacă registrul tampon de transmitere devine gol; emiterea acestei comenzi împiedică alte întreruperi la transmisie pînă la încărcarea următorului caracter în tamponul de transmisie sau pînă ce s-a trimis complet caracterul de redundanță ciclică (CRC).
  - comanda 6 — inițializare erori; inițializează bistabilele care înregistrează erorile; erorile de paritate și de depășire de viteză sînt înregistrate în registrul RR1 pînă cînd sînt șterse cu această comandă; astfel, erorile de paritate care apar la transferurile de bloc pot fi examinate la sfîrșitul blocului.
  - comanda 7 — revenire din întrerupere; trebuie emisă pentru canalul A și este interpretată de SIO ca și o comandă RETI de pe magistrala de date; șterge bistabilul de întrerupere în servire a celui mai prioritar dispozitiv intern în servire și permite astfel întreruperi provenind de la dispozitive mai puțin prioritare din lanțul de întreruperi; această comandă permite utilizarea lanțului de priorități intern și în sisteme fără lanț de priorități extern sau comandă RETI;
- D6 — D7 — sînt codurile de inițializare CRC; codul nul nu are nici un efect; comanda de inițializare a generatorului CRC la transmisie îi aduce toți biții la valoarea 0, în afară de modul SDLC, la selectarea căruia inițializarea se face la valoarea 1; vericatorul CRC este de asemenea inițializat la valoarea 1 a biților în modul SDLC.
- Registrul WR1 — stabilește modurile de tratare a întreruperilor și rolul funcției WAIT/READY.
- D0 — validarea întreruperilor de stare/externe; permite întreruperi la tranzițiile care apar pe intrările DCD, CTS sau SYNC, la selecția începerii sau terminării semnalului de Pauză/Evitare, sau la începerea transmiterii caracterului CRC sau de sincronizare cînd bistabilul „Transmisie sub viteză/Sfîrșit de mesaj” este înscris.
- D1 — bitul de validare a întreruperilor la transmisie; permite apariția întreruperilor ori de cîte ori registrul tampon de transmisie devine gol.
- D2 — starea afectează vectorul de întreruperi; se referă doar la canalul B; dacă D2=0, vectorul de întreruperi înscris în WR2 este transmis unității centrale în secvența de recunoaștere a întreruperii; dacă D2=1, vectorul de întreruperi transmis unității centrale va avea cîmpul V3, V2, V1 variabil, în funcție de următoarele condiții de întrerupere, ca mai jos, unde sînt marcate cu „\*” condițiile speciale de recepție: eroare de paritate, eroare de depășire de viteză la recepție, eroare de încadrare, sfîrșit de cadru (SDLC):

	V3	V2	V1	Condiția de întrerupere
Canal B	0	0	0	Registrul tampon de transmisie gol
	0	0	1	Schimbare de stare/externă
	0	1	0	Caracter de recepție disponibil
	0	1	1	Condiție specială de recepție*
Canal A	1	0	0	Registrul tampon de transmisie gol
	1	0	1	Schimbare de stare/externă
	1	1	0	Caracter de recepție disponibil
	1	1	1	Condiție specială de recepție*

- D3, D4 — selectează modurile de întrerupere la recepție; în oricare dintre modurile de întrerupere la recepție, o condiție specială la recepție poate determina o întrerupere și poate modifica vectorul de întrerupere; pentru modul de întrerupere



pere la recepția tuturor caracterelor cu  $D4=1$  și  $D3=0$ , eroarea de paritate este o condiție specială de recepție, ceea ce nu este valabil dacă  $D4=1$  și  $D3=1$ .  
 $D5 - D7$  — selectează funcția  $\overline{WAIT}/\overline{READY}$ ;  $D7=1$  validează funcția;  $D6=1$  alege funcția  $\overline{READY}$ , caz în care ieșirea  $\overline{WAIT}/\overline{READY}$  trece de la 1 la 0 când circuitul Z80 SIO este gata să transfere date; dacă  $D6=0$ , se selectează funcția  $\overline{WAIT}$  și ieșirea  $\overline{WAIT}/\overline{READY}$  este în starea cu drenă în gol și trece la 0 când este activă; funcțiile  $\overline{WAIT}$  și  $\overline{READY}$  pot fi utilizate atât în modul de transmisie cât și în cel de recepție, dar nu în ambele simultan; dacă  $D5=1$ , funcția  $\overline{WAIT}/\overline{READY}$  răspunde la condiția registrului tampon de recepție (plin sau gol); dacă  $D5=0$ , funcția răspunde la condiția registrului tampon de transmisie (gol sau plin); starea logică a ieșirii  $\overline{WAIT}/\overline{READY}$ , când este activă sau inactivă, depinde de combinația de moduri selectate:

$D7=0, D6=1$   $\overline{READY}=1$

$D7=0, D6=0$   $\overline{WAIT}$  este flotantă

$D7=1, D5=0$   $\overline{READY}=1$  când registrul tampon de transmisie este plin;  $\overline{WAIT}=0$  când tamponul de transmisie este plin și portul de date SIO este selectat;  $\overline{READY}=0$  când tamponul de recepție este gol;  $\overline{WAIT}$  este flotantă când tamponul de transmisie este gol.

$D7=1, D5=1$   $\overline{READY}=1$  când tamponul de recepție este gol;  $\overline{WAIT}=0$  când tamponul de recepție este gol și portul de date SIO este selectat;  $\overline{READY}=0$  când tamponul de recepție este plin;  $\overline{WAIT}$  este flotantă când tamponul de recepție este plin.

Funcția  $\overline{READY}$  poate să apară oricând circuitul Z80 SIO nu este selectat; când ieșirea  $\overline{READY}$  devine activă, circuitul DMA emite un semnal  $\overline{IORQ}$  și fixează intrările  $B/\overline{A}$  și  $C/\overline{D}$  ale lui Z80 SIO pentru a transfera date; ieșirea  $\overline{READY}$  devine inactivă când  $\overline{IORQ}$  și  $\overline{CS}$  devin active; funcția  $\overline{READY}$  apare intern în circuitul Z80 SIO, fie că este sau nu adresat și ca urmare ieșirea  $\overline{READY}$  devine inactivă când are loc orice transfer de date sau comenzi cu unitatea centrală; nu apar erori, deoarece controlerul DMA nu este validat când are loc transferul cu unitatea centrală; pe de altă parte, funcția  $\overline{WAIT}$  este activă numai dacă unitatea centrală încearcă să citească din circuitul Z80 SIO date care nu au fost încă recepționate, ceea ce apare frecvent când se utilizează instrucțiuni de transfer de bloc; funcția  $\overline{WAIT}$  mai poate deveni activă sub controlul programului, dacă unitatea centrală încearcă să înscrie date în Z80 SIO, în timp ce registrul tampon de transmisie este încă plin; faptul că ieșirea  $\overline{WAIT}$  a fiecărui canal poate deveni activă când este adresat celălalt canal (deoarece este adresat circuitul Z80 SIO), nu afectează funcționarea buclelor de program sau a instrucțiunilor de transfer de bloc.

Registrul WR2 — este un registru pentru vectorul de întrerupere și există doar în canalul B;  $V7 \div V4$  și  $V0$  sînt returnați spre unitatea centrală așa cum au fost înscrisi;  $V3 \div V1$  sînt returnați cum au fost înscrisi dacă bitul din D2 WR1 (Starea afectează vectorul) este 0; dacă este 1, biții  $V3 \div V1$  sînt modificați cum s-a indicat la descrierea lui D2 din WR1.

Registrul WR3 — conține parametri și biți de control ai logicii de recepție:  
 $D0$  — permite începerea recepției, avînd rol de bit de validare a receptorului, dacă  $D0=1$ ; trebuie înscris numai după ce toți ceilalți parametri de recepție sînt fixați și receptorul este complet inițializat.

$D1$  — inhibarea încărcării caracterului de sincronizare; permite să nu se încarce în registrele tampon de recepție, caracterele de sincronizare conducătoare, care



preced mesajul, dacă  $D1=1$ ; deoarece calcularea CRC nu este oprită prin înlăturarea caracterelor de sincronizare, această posibilitate trebuie folosită numai la începutul mesajului.

- D2 — mod de căutare de adresă; dacă  $D2=1$ , permite respingerea mesajelor, în modul SDLC, ale căror adrese nu corespund nici cu adresa programată în WR6, nici cu adresa globală 11111111; ca urmare, nu apar întreruperi la recepție în modul de căutare de adresă, decât dacă se găsește o coincidență de adrese.
- D3 — validare CRC la recepție; dacă  $D3=1$ , determină începerea (sau reînceperea) calculului CRC la începutul ultimului caracter transferat din registrul de deplasare la recepție spre stiva tampon, indiferent de numărul caracterelor din stivă; detalii se dau la descrierea căutării CRC la recepția SDLC și la aceea a căutării de erori CRC pentru recepția sincronă.

- D4 — intrare în faza de urmărire; permite reintrarea în faza de urmărire dacă s-a pierdut sincronizarea caracterelor, în modul sincron, sau dacă, în modul SDLC, conținutul unui mesaj care este recepționat, nu este necesar; circuitul Z80 SIO intră automat în faza de urmărire după o inițializare; reintrarea se face prin  $D4=1$ , ceea ce înscrie și bitul Sync/Hunt (D4) din registrul RR0.

- D5 — autovalidări; dacă este selectat acest mod,  $\overline{DCD}$  și  $\overline{CTS}$  devin intrările de validare a receptorului și respectiv transmițătorului (când  $D5=1$ ); dacă  $D5=0$ ,  $\overline{DCD}$  și  $\overline{CTS}$  sînt simple intrări pentru biții de stare care le corespund în registrul RR0.

- D6, D7 — număr de biți/caracter la recepție; determină numărul de biți primiți în serie la recepție, din care se assemblează un caracter; ambii pot fi schimbați în timpul asamblării unui caracter, dar trebuie să fie schimbați înainte de a se fi atins numărul de biți programat anterior.

Registrul WR4 — conține biți de control care afectează atât receptorul cât și transmițătorul; în rutina de inițializare a transmisiei și recepției, acești biți trebuie înscriși înainte de a emite WR1, WR3, WR5, WR6, WR7:

- D0 — paritate; dacă  $D0=1$ , un bit adițional față de cei stabiliți la număr de biți/caracter, este transmis și este de asemenea așteptat la recepție; în modul de recepție, bitul de paritate recepționat este transferat spre unitatea centrală ca parte din caracter, cu excepția cazului în care s-au ales 8 biți/caracter.

- D1 — paritate pară/impară; dacă se lucrează cu paritate, acest bit stabilește dacă bitul de paritate este transmis și verificat la recepție ca par sau impar ( $D1=1$  pentru paritate pară).

- D2, D3 — numărul de biți de stop; stabilește numărul biților de stop adăugați la caracterele asincrone; receptorul caută întotdeauna un bit de stop; codul  $D3=D2=0$  arată că se va selecta un mod sincron.

- D4, D5 — moduri de sincronizare; aleg între diferitele opțiuni pentru caracterele de sincronizare.

- D6, D7 — modul pentru frecvența de tact; specifică factorul cu care se divizează frecvența de tact  $\overline{TXC}$  și  $\overline{RXC}$  pentru a obține viteza de transmisie, respectiv recepție; pentru modurile sincrone, trebuie ales factorul 1 ( $D7=D6=0$ ); pentru modurile asincrone, se poate alege orice factor, cu restricția de a avea același factor pentru receptor și pentru transmițător; frecvența de tact trebuie să fie cel puțin de 5 ori mai mare decât frecvența de schimb a datelor, în toate modurile; dacă se selectează factorul 1, sincronizarea biților trebuie realizată extern.

Registrul WR5 — conține biți care afectează numai transmițătorul, cu excepția lui D2, care afectează și receptorul.

- D0 — validarea CRC la transmisie; acest bit determină dacă CRC este calculat pentru un anumit caracter la transmisie; dacă  $D0=1$ , codul CRC este calculat pentru acest caracter, cînd este încărcat din registrul tampon de transmisie în registrul de deplasare la transmisie; codul CRC nu este trimis automat decât dacă acest bit este 1, cînd există condiția de transmisie sub viteză (transmit underrun).



- D1 — cerere de transmisie; este bitul de control pentru pinul  $\overline{\text{RTS}}$  al SIO; cînd  $D1=1$  semnalul  $\overline{\text{RTS}}$  trece la 0; cînd  $D1=0$ ,  $\overline{\text{RTS}}$  trece la 1; în modul asincron,  $\overline{\text{RTS}}$  trece la 1 numai după ce toți biții caracterului au fost transmiși și registrul tampon de transmisie este gol; în modurile sincrone, semnalul  $\overline{\text{RTS}}$  urmărește starea bitului RTS (bitul D1).
- D2 — CRC — 16/SDLC; selectează polinomul CRC utilizat atât de transmițător cît și de receptor; cînd  $D2=1$ , este utilizat polinomul CRC-16 ( $X^{16}+X^{15}+X^2+1$ ); cînd  $D2=0$ , se utilizează polinomul SDLC ( $X^{16}+X^{12}+X^5+1$ ) dacă este selectat modul SDLC, generatorul și vericatorul CRC sînt înscrisi inițial cu biți egali cu 1 și se utilizează o secvență specială de verificare; polinomul CRC din modul SDLC trebuie să fie selectat cînd se selectează modul SDLC; dacă nu se selectează modul SDLC, generatorul și vericatorul CRC sînt înscrisi inițial cu cifre zero, pentru ambele polinoame.
- D3 — validarea transmițătorului; datele nu sînt transmise pînă cînd  $D3=1$  și ieșirea de transmisie de date este menținută la nivelul de marcare; datele sau caracterele de sincronizare în curs de transmitere sînt transmise complet dacă bitul D3 este adus la 0 după începerea transmisiei; dacă transmisia este dezactivată în timpul transmiterii unui cod CRC, se transmit caractere de sincronizare sau indicatori în locul codului CRC.
- D4 — pauză de trimitere (send break); cînd  $D4=1$ , se aduce ieșirea de transmisie a datelor, TXD la condiția de spațiere, indiferent de datele care se transmit; cînd  $D4=0$  linia TXD revine la nivelul de marcare.
- D5, D6 — număr de biți/caracter la transmisie; fixează numărul de biți din care se assemblează fiecare caracter la transmisie; primul bit transmis este cel mai puțin semnificativ; în modul  $D5=D6=0$ , se pot transmite 1 pînă la 5 biți/caracter; unitatea centrală își va forma caracterul astfel:

Număr biți/ caracter	D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	0	0	0	D
2	1	1	1	0	0	0	D	D
3	1	1	0	0	0	D	D	D
4	1	0	0	0	D	D	D	D
5	0	0	0	D	D	D	D	D

- D7 — terminalul de date gata; este bitul de control pentru semnalul  $\overline{\text{DTR}}$  al circuitului Z80 SIO; cînd  $D7=1$ ,  $\overline{\text{DTR}}$  devine activ, trecînd la 0; cînd  $D7=0$ ,  $\overline{\text{DTR}}$  este inactiv, trecînd la 1.

Registrul WR6 — este programat pentru a conține caracterul de sincronizare la transmisie în modul Monosync, primii 8 biți dintr-un caracter de sincronizare de 16 biți în modul Bisync, sau un caracter de sincronizare la transmisie în modul de sincronizare externă; în modul SDLC, este programat pentru a conține cîmpul de adresă secundar utilizat pentru a fi comparat cu cîmpul de adresă al cadrului SDLC.

Registrul WR7 — este programat pentru a conține caracterul de sincronizare la recepție în modul Monosync, un al doilea octet (ultimii 8 biți) ai unui caracter de sincronizare de 16 biți în modul Bisync și un indicator (01111110), în modul SDLC; registrul de scriere WR7 nu este utilizat în modul cu sincronizare externă.

### Funcționarea la întreruperi a circuitului Z80 SIO

După primirea unui semnal de cerere de întrerupere de la un circuit SIO, ( $\overline{\text{INT}}$  adus la 0), unitatea centrală trimite o secvență de recunoaștere a întreruperii ( $\overline{\text{MI}}=0$  și  $\overline{\text{IORQ}}=0$ ). Circuitul SIO conține o structură internă de priorități la întrerupere



pentru a putea trata întreruperi suprapuse, pentru diferitele funcții ale celor două canale. Această structură poate fi utilizată într-un lanț de priorități extern, de către utilizator, conținând mai multe circuite periferice. Intrarea IEI a celui mai prioritar circuit este la 1. Un dispozitiv care așteaptă servirea unei întreruperi sau este în curs de servire, își aduce ieșirea IEO la 0. Pentru dispozitivele care nu așteaptă servirea sau nu sînt în curs de servire,  $IEO = IEI$ . Pentru a asigura condiții stabile în lanțul de priorități, toate semnalele de stare a întreruperii trebuie să fie stabile în timp ce  $\overline{M1} = 0$ . Cînd  $IORQ = 0$ , dispozitivul cu cea mai mare prioritate care a cerut întreruperea plasează vectorul de întrerupere pe magistrala de date și își înscrie bistabilul intern de „întrerupere în servire”.

Pentru revenirea din întrerupere, unitatea centrală Z80 emite în mod obișnuit o instrucțiune RETI (de revenire din întrerupere) la sfîrșitul unei rutine de servire a întreruperii. Codul ED4D<sub>H</sub> al instrucțiunii inițializează bistabilul de întrerupere în curs de servire din SIO pentru a termina întreruperea care a fost servită. Aceasta se realizează în lanțul de întreruperi așa cum se descrie în continuare.

Funcționarea obișnuită a lanțului de priorități poate fi utilizată pentru a detecta o întrerupere în așteptare. Totuși, el nu poate distinge o întrerupere în servire de una în așteptare, nerecunoscută, cu o prioritate mai mare. Cînd este decodificat codul ED<sub>H</sub>, lanțul de priorități este modificat, forțînd IEO la 1, la toate circuitele care au cerut întreruperi ce nu au fost recunoscute. Astfel, lanțul de priorități identifică dispozitivul care este în curs de servire, ca fiind singurul cu  $IEI = 1$  și  $IEO = 0$ . Dacă următorul octet cod de operație este 4D<sub>H</sub>, bistabilul de întrerupere în curs de servire este inițializat (șters).

Timpul de propagare prin lanțul de priorități la întreruperi (atît pentru tranzițiile 1→0 cît și pentru 0→1) limitează numărul de dispozitive care pot fi plasate într-un astfel de lanț. Timpul de propagare poate fi îmbunătățit prin calculul în avans al transportului (carry look ahead), sau prin extinderea ciclului de recunoaștere a întreruperii.

Tratarea întreruperilor suprapuse este ilustrată în figura 5.9.

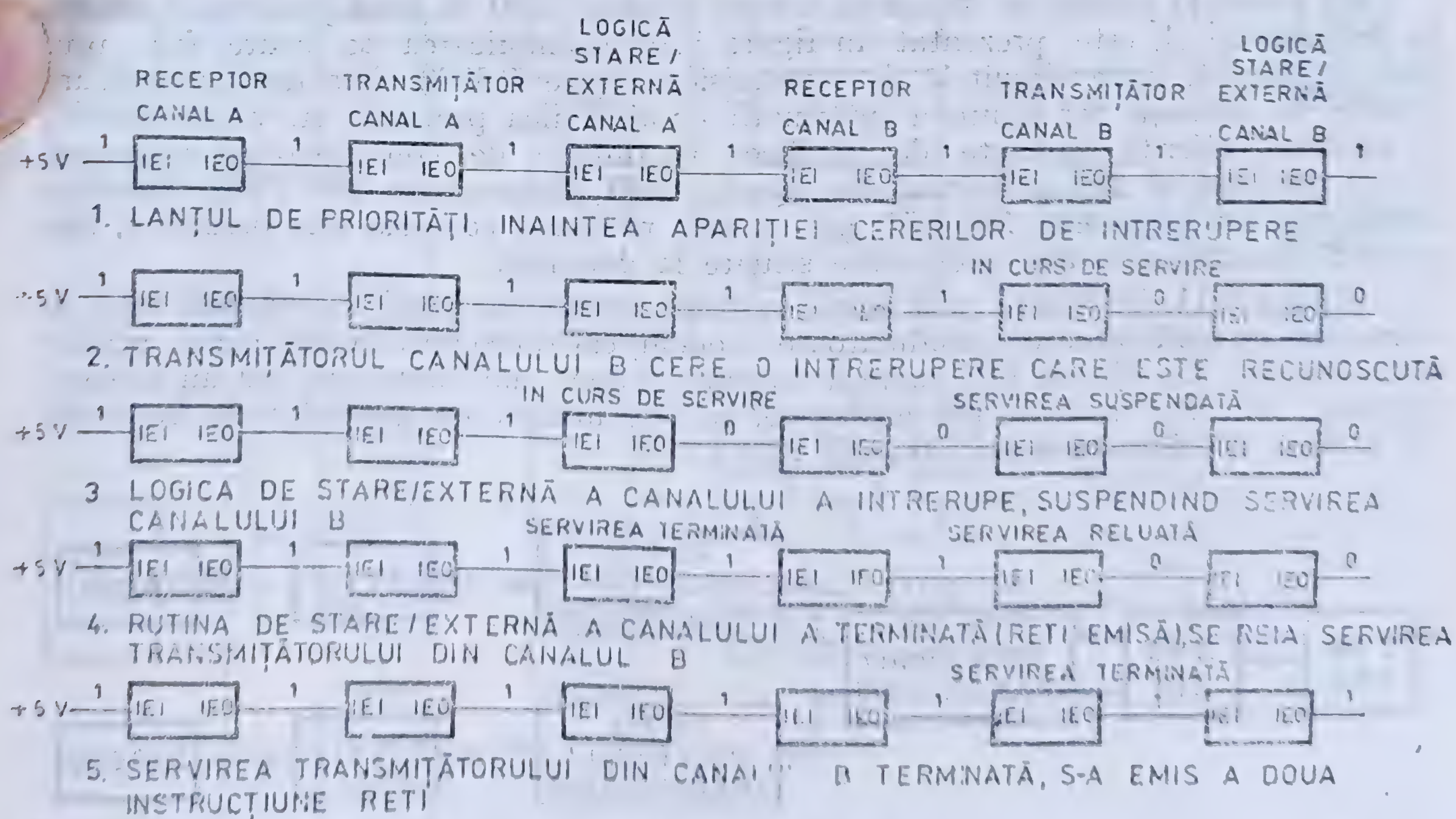


Fig. 5.9 Secvență tipică de întreruperi suprapuse.



## Funcționarea asincronă

Pentru transmisie asincronă, circuitul Z80 SIO trebuie inițializat cu lungimea de caracter, viteza de schimb a informației, numărul de biți de stop, paritate pară sau impară, modul de întrerupere și validarea receptorului sau transmițătorului. Acești parametri sînt încărcăți pe programul sistemului, în ordinea de înscriere: WR4, WR1, WR3, WR5 (WR4 înaintea celorlalte registre).

Dacă datele se transmit printr-un modem sau o interfață RS232, ieșirile  $\overline{\text{RTS}}$  și  $\overline{\text{DTR}}$  trebuie fixate împreună cu bitul de validare a transmisiei, fără de care nu poate începe transmisia.

Posibilitatea autovalidărilor permite programatorului să trimită primul caracter (dată) al mesajului spre Z80 SIO, fără să aștepte semnalul  $\overline{\text{CTS}}$ . Dacă bitul de autovalidări este înscris, SIO va aștepta pînă cînd  $\overline{\text{CTS}}$  trece la 0 înainte să transmită data.  $\overline{\text{CTS}}$ ,  $\overline{\text{DCD}}$  și  $\overline{\text{SYNC}}$  sînt linii generale de I/E care pot fi folosite și pentru alte funcții. Dacă se utilizează  $\overline{\text{CTS}}$  în alt scop, bitul de autovalidări trebuie programat la 0.

La înscrierea cuvintelor în registrele de scriere, pentru modul asincron, există următoarele restricții:

- în WR3: D4=0, D3=0, D2=0, D1=0
- în WR4: D5=0, D4=0,
- în WR5: D2=0, D0=0

Registrul WR2 (numai în canalul B) conține vectorul de întreruperi și WR1 definește modul de întrerupere și modul de transfer de date. WR6 și WR7 nu se utilizează în modurile asincrone.

## 5.2. APLICAȚII ALE CIRCUITULUI Z80 SIO

### 1. Comunicații sincrone/asincrone de la procesor la procesor, pe o singură linie

Un exemplu posibil de conectare a unui procesor Z80 la două procesoare situate la distanță de el, este prezentat în figura 5.10. Comunicarea se poate face prin linia telefonică, între circuitele de transmisie/recepție RS232. Cele două procesoare situate la distanță pot schimba informații cu al treilea procesor, cu viteze diferite și utilizînd diferite variante de protocol. În funcție de complexitatea aplicației, pot fi necesare și alte periferice din familia Z80 (de exemplu Z80 CTC). Canalele neutilizate ale circuitelor Z80 SIO pot fi utile pentru a controla alte periferice, sau pentru conectarea la alte procesoare situate la distanță.

Figura 5.11 ilustrează utilizarea ambelor canale ale unui circuit Z80 SIO în conexiune cu modulate/demodulate (modem) care au și opțiunea de canal primar sau secundar. Un circuit SIO poate fi conectat la două modeme care nu au această opțiune. În cazul modemurilor asincrone, trebuie utilizat un generator de frecvență de schimb a informației (de exemplu Z80 CTC).

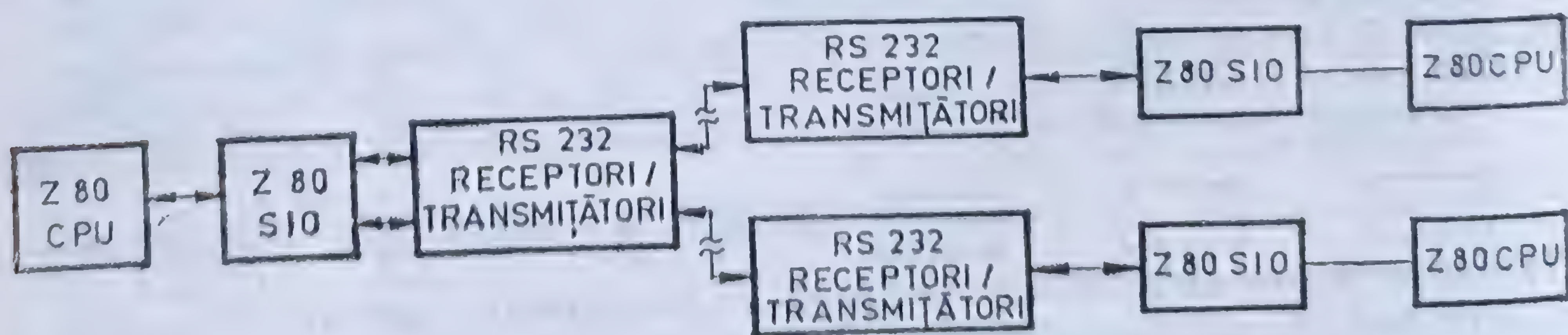


Fig. 5.10 Comunicații sincrone/asincrone de la procesor la procesor, pe o linie telefonică.



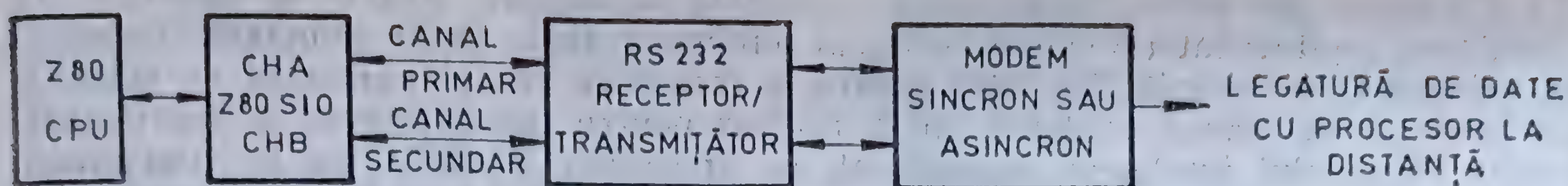


Fig. 5.11 Utilizarea ambelor canale ale unui circuit Z80 SIO.

## 2. Echipament de achiziție de date

Un echipament de achiziție de date, relativ complex, utilizând două circuite Z80 SIO, și care poate realiza o varietate de funcții, este reprezentat în figura 5.12. Echipamentul poate fi utilizat pentru colectarea de date de la mai multe terminale, pe linii de joasă viteză și pentru transmiterea lor pe o singură linie de mare viteză, după editarea și reformatarea lor.

Circuitul controler de acces direct la memorie Z80 DMA este utilizat împreună cu circuitul Z80 SIO 2 pentru a transmite datele reformatate, la viteză mare, cu protocolul necesar. Semnalul de tact pentru transmisie pe acest canal este asigurat de modemul de mare viteză.

Circuitul Z80 CTC asigură semnalele de tact pentru transmisia și recepția pe liniile de joasă viteză și este de asemenea utilizat ca numărător care măsoară intervalele de timp pentru alte funcții.

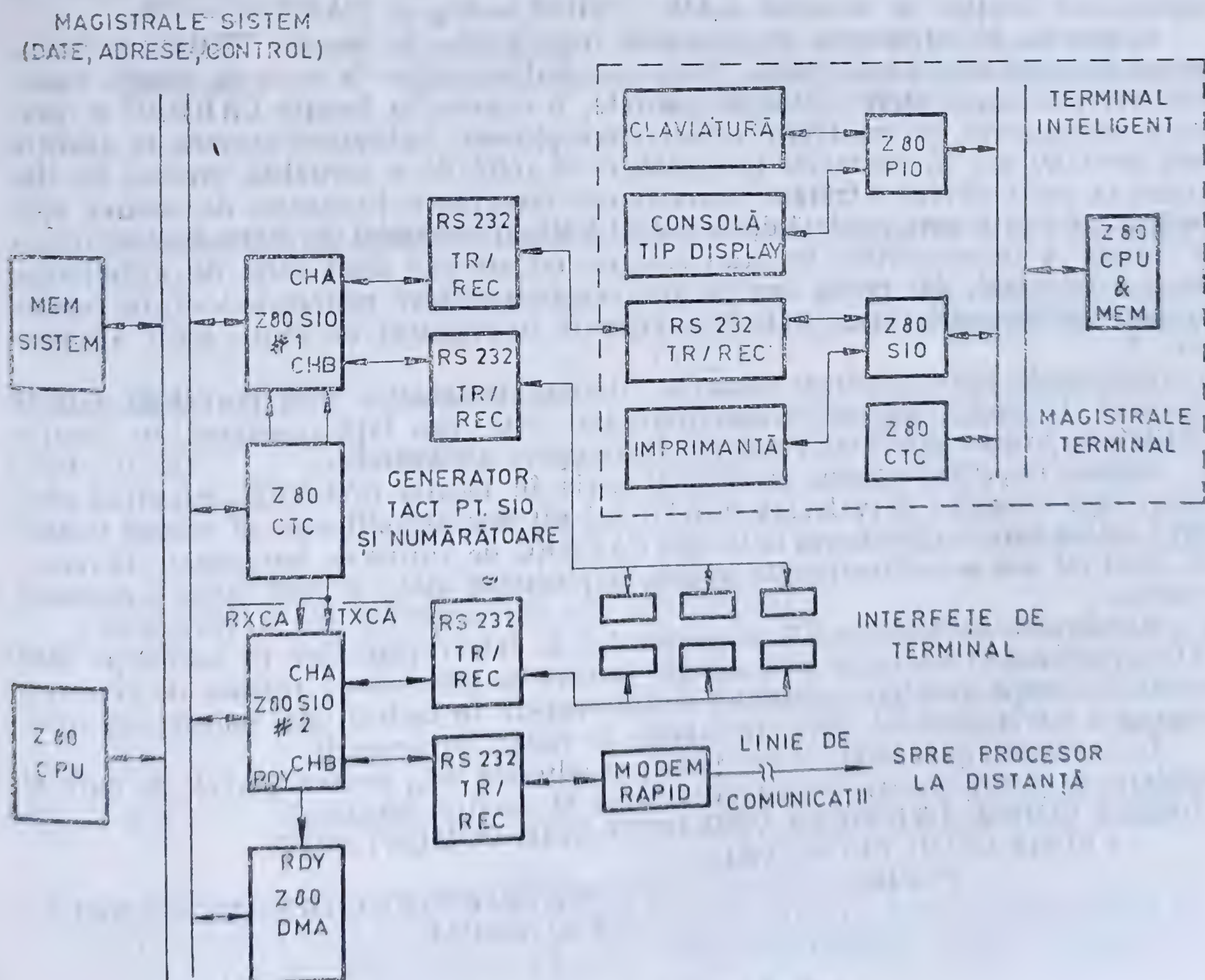


Fig. 5.12 Echipament de achiziție de date.



Circuitul Z80 SIO 1 controlează terminale locale sau situate la distanță. În figură este reprezentat un singur terminal inteligent, între liniile punctate. Terminalul utilizează un circuit Z80 SIO pentru a comunica cu echipamentul de achiziție de date pe un canal, al doilea fiind utilizat pentru interfațare cu o imprimantă. Pentru terminalul inteligent reprezentat, se presupune posibilitatea de funcționare interactivă cu operatorul. În funcție de posibilitățile soft și hard ale sistemului, echipamentul de achiziție de date poate utiliza diferite metode (de exemplu depozitează și avansează sau menține și avansează) pentru controlul transferului de informație între terminalele lente și procesorul rapid situat la distanță. Dacă se prevede canalul de mare viteză cu posibilitatea de decuplare, canalul poate fi conectat la un număr de procesoare situate la distanță printr-o linie care se poate comuta.

### 3. Utilizarea circuitului Z80 SIO pentru comunicație asincronă, cu validarea întreruperilor

În continuare este prezentat un exemplu de programare a circuitului Z80 SIO. Programul principal apelează subrutina INISIO pentru inițializarea circuitului și subrutinele SCRCAR și CITCAR, ori de câte ori este necesar, pentru transmisia și respectiv recepția de date pe canalul B din SIO.

Programul principal înscrie în registrul I octetul superior al adresei tabelului care conține adresele subrutinelor de întrerupere, FA<sub>H</sub> și apelează subrutina de inițializare a circuitului Z80 SIO, care înscrie cuvinte de comandă în registrele de scriere WR2, WR4, WR5, WR3 și WR1, printr-o singură instrucțiune, OTIR și inițializează locațiile de memorie RAM, (TRGOL)=00<sub>H</sub> și (CARREC)=FF<sub>H</sub>.

Subrutina de întrerupere la transmisie încarcă 00<sub>H</sub> în locația TRGOL și inițializează întreruperile la transmisie. Subrutina de întrerupere la recepție citește caracterul recepționat, îi șterge bitul de paritate, îl depune în locația CARREC și validează întreruperea pe următorul caracter recepționat. Subrutina apelată la apariția unei erori nu are în exemplul prezentat decât rolul de a inițializa erorile. În alte cazuri ea poate efectua o tratare a caracterului recepționat în funcție de eroarea apărută, a cărei citire este posibilă din registrul RR1 al canalului B. Asemănător, rutina de tratare a întreruperilor de stare/externe nu are aici decât rolul de a inițializa aceste întreruperi, dar poate face în alte cazuri o tratare mai complexă, în funcție de cauza întreruperii, care poate fi examinată în registrul de citire RR0 al canalului B.

Rutina de scriere a unui caracter trimite un caracter prin portul de date al canalului B, dacă tamponul transmițătorului este gol, fapt constatat în locația TRGOL, ca urmare a acțiunii rutinei de întrerupere la transmisie.

Rutina de citire a unui caracter îl preia în locația CARREC, așteptând până apare, fapt semnalat de valoarea 0 a bitului cel mai semnificativ al acestei locații. După citirea caracterului depus în locația CARREC de rutina de întrerupere la recepție, bitul cel mai semnificativ este înscris, împiedicând astfel o nouă citire a aceluiași caracter.

Modificarea biților D3—D1 ai vectorului de întrerupere face ca adresa pe care CPU o formează pentru a găsi adresa la care se află adresa rutinei de tratare a întreruperii după primirea vectorului de întrerupere în cadrul unei secvențe de recunoaștere a întreruperii, să difere în funcție de cauza întreruperii.

În exemplul prezentat, s-au presupus adresele 81<sub>H</sub> pentru portul de date al canalului B și 83<sub>H</sub> pentru portul de control al acestui canal.

ADRESA CODUL ETICHETA COD OPERATIE COMENTARIU

PROGRAMUL PRINCIPAL

PRPR: ...

; INSTRUCTIUNI DIN PROGRAMUL  
; PRINCIPAL

.  
. .  
.



3EFA  
ED47

LD A,OSTI  
LD I,A

; OCTET SUPERIOR AL ADRESEI TA-  
BELULUI DE ADRESE ALE SUBRU-  
TINELOR DE INTRERUPERE. IN  
REGISTRUL I (OSTI=FAH,OITI=  
=00H)

CD2340

CALL INISIO

; APELEAZA SUBROUTINA DE INITIA-  
LIZARE

CD4840

CALL CITCAR

; APELEAZA PENTRU CITIRE CARAC-  
TER

3EXX

CD3A40

LD A,CAR1

CALL SCRCAR

; INCARCA UN CHARACTER IN A  
; APELEAZA PENTRU SCRIERE CA-  
; RACTER LA CONSOLA

3EXX

CD3A40

LD A,CAR2

CALL SCRCAR

; INCARCA UN CHARACTER IN A  
; APELEAZA PENTRU SCRIERE CA-  
; RACTER LA CONSOLA

; TABEL CU ADRESELE SUBROUTINELOR DE TRATARE A INTRERUPE-  
RILOR

FA00 0040

TASI: DW INTTRS

; ADRESA RUTINEI DE INTRERU-  
; PERE LA TRANSMISIE PE CANA-  
; LUL B

FA02 1E40

DW INTSE

; ADRESA RUTINEI DE INTRERUPE-  
; RE DE STARE/EXTERNA

FA04 0D40

DW INTREC

; ADRESA RUTINEI DE INTRERU-  
; PERE LA RECEPTIE PE CANALUL B

FA06 1940

DW INTER

; ADRESA RUTINEI DE INTRERUPE-  
; RE LA CONDITIE SPECIALA DE  
; RECEPTIE

; LOCATII RAM UTILIZATE DE PROGRAM

0000

XX

TRGOL:

; LOCATIE RAM PENTRU INDICAREA  
; STarii TRANSMITATORULUI;  
; S-A TRANSMIS CHARACTERUL SAU  
; NU EXISTA CHARACTER DE TRANS-  
; MIS DACA (TRGOL)=00H

0001

XX

CARREC:

; LOCATIE RAM PENTRU DEPOZI-  
; TAREA UNUI CHARACTER RECEP-  
; TIONAT

; SUBROUTINE DE TRATARE A INTRERUPERILOR:

; SUBROUTINA DE TRATARE A INTRERUPERILOR LA TRANSMISIE PE  
; CANALUL B; SE EXECUTA LA TRANSMITEREA UNUI CHARACTER  
; SI FACE (TRGOL) = 0



```

4000 F5 INTTRS: PUSH AF ; SALVEAZA AF
4001 AF NOR A ; STERGE A
4002 320000 LD (TRGOL),A ; INTRODUC 0 IN LOCATIA (TRGOL)
4005 3E28 LD A,00101000B ; INITIALIZARE INTRERUPERI DE
; TRANSMISIE
4007 D383 SFINT: OUT (SIOBC),A ; INSCRIE CUVINT DE COMANDA
; IN SIO, CANAL B, CONTROL
4009 F1 POP AF ; REFACE AF
400A FB EI ; VALIDEAZA INTRERUPERILE
400B ED4D RETI ; REVINE DIN INTRERUPERE
; SUBROUTINA DE TRATARE A INTRERUPERILOR LA RECEPTIE PE
; CANALUL B; SE EXECUTA LA RECEPTIA UNUI CHARACTER, PE CARE
; IL DEPUNE IN LOCATIA DE MEMORIE RAM CU ADRESA CARREC
400D F5 INTREC: PUSH AF ; SALVEAZA AF
400E DB81 IN A,(SIOBD) ; CITESTE CHARACTER IN A
4010 E67F AND 7FH ; STERGE BITUL DE PARITATE
4012 320100 LD (CARREC),A ; DEPUNE CHARACTER IN LOCATIA
; CU ADRESA CARREC
4015 3E20 LD A,00100000B ; CUVINT VALIDARE INTRERUPERI
; PE URMATORUL CHARACTER RE-
; CEPTIONAT, SE VA INSCRIE IN
; WR0
4017 18EE JR SFINT ; SALT PENTRU REVENIRE DIN INT
; SUBROUTINA DE TRATARE A INTRERUPERILOR LA CONDITII SPE-
; CIALE DE RECEPTIE
4019 F5 INTER: PUSH AF ; SALVEAZA AF
401A 3E30 LD A,001100C0B ; CUVINT DE STERGERE ERORI
; (INITIALIZARE ERORI), SE VA IN-
; SCRIE IN WR0
401C 18FB JR SFINT ; REVINE, FARA ALTE DECIZII
; SUBROUTINA DE TRATARE A INTRERUPERILOR DE STARE/EXTERNE
401E F5 INTSE: PUSH AF ; SALVEAZA AF
401E 3E10 LD A,00010000B ; COMANDA INITIALIZARE INTRE-
; RUPERI DE STARE/EXTERNE
4021 18E4 JR SFINT ; REVINE, FARA ALTE DECIZII
; SUBROUTINA DE INITIALIZARE SIO; INITIALIZEAZA CANALUL B
; PENTRU TRANSMISIE ASINCRONA
4023 F5 INISIO: PUSH AF ; SALVEAZA AF
4024 C5 PUSH BC ; SALVEAZA BC
4025 E5 PUSH HL ; SALVEAZA HL
4026 210044 LD HL,SIOTBL ; ADRESA TABELULUI CU CUVIN-
; TE DE COMANDA PENTRU SIO, IN
; HL
4029 01830A LD BC,0A83H ; LUNGIMEA TABELULUI CUVINTE-
; LOR DE COMANDA, OAH IN B SI
; ADRESA PORTULUI DE CONTROL
; PENTRU CANAL B SIO,83H, IN C
402C EDB3 OTIR ; INSCRIE CONTINUT TABEL IN SIO
402E AF XOR A ; STERGE A
402F 320000 LD (TRGOL),A ; INTRODUC 0 IN LOCATIA TRGOL
4032 3D DEC A ; INTRODUC FFH IN A
4033 320100 LD (CARREC),A ; INTRODUC FFH LA ADRESA
; CARREC

```



```

4036 E1 POP HL ; REFACE HL,
4037 C1 POP BC ; REFACE BC
4038 F1 POP AF ; REFACE AF
4039 C9 RET ; REVINE
;SUBROUTINA DE SCRIERE LA CONSOLA A UNUI CARACTER DIN A
403A F5 SCRCAR: PUSH AF ; SALVEAZA AF
403B 3A0000 ASTTRS:
LD A,(TRGOL); CONTINUTUL LUI TRGOL IN A
AND A ; TESTEAZA DACA ESTE 0
403E A7 JR NZ,ASTTRS; ASTEAPTA PINA LA TRANSMITE-
403F 20F9 ; REA CARACTERULUI ANTERIOR,
; DUPA CARE (TRGOL)=0
4041 F1 POP AF ; REFACE AF DACA A=0
4042 320000 LD (TRGOL),A; INTRODUCHE CARACTERUL IN LO-
; CATIA TRGOL, PENTRU CA SA NU
; MAI CONTINUA 00H
4045 D381 OUT (SIOBD),A; INSCRIE CARACTERUL IN POR-
; TUL B,DATE DIN SIO
4047 C9 RET ; REVINE
;SUBROUTINA DE CITIRE A UNUI CARACTER DE LA CONSOLA IN
;REGISTRUL A
4048 E5 CITCAR: PUSH HL ; SALVEAZA HL
4049 210100 LD HL,CARREC; ADRESA DE DEPUNERE IN RAM,
; IN HL
404C 7E ASTREC: LD A,(HL) ; CARACTERUL DIN CARREC IN A
404D B7 OR A ; POZITIONEAZA BITUL S
404E FA4C40 JP M,ASTREC; ASTEAPTA DACA D7=1, CIND NU
; EXISTA CARACTER RECEPTIONAT
4051 CBFE SET 7,(HL) ; INTRODUCHE 1 IN BITUL D7 DIN
; LOCATIA CU ADRESA CARREC
4053 E1 POP HL ; REFACE HL
4054 C9 RET ; REVINE
;TABELUL CUVINTELOR DE COMANDA PENTRU PROGRAMAREA
;CIRCUITULUI SIO;
;CUVINTELE SINT INSCRISE IN SIO DE SUBROUTINA INISIO
4400 00000010 SIOTBL: DEFB 02H ; INDICA REGISTRUL 2,SE INSCRIE
; IN REGISTRUL WR0
4401 00000000 DEFB 00H ; VECTORUL DE INTRERUPERI
; (OITI); SE VA INSCRIE IN WR2
4402 00000100 DEFB 04H ; INDICA WR4, SE INSCRIE IN WR0
4403 01000111 DEFB 47H ; FIXEAZA DIVIZAREA LUI TNC,RNC
; CU 16, MOD ASINCRON, UN BIT
; DE STOP, PARITATE PARA; SE
; INSCRIE IN WR4
4404 00000101 DEFB 05H ; INDICA WR5, SE INSCRIE IN WR0
4405 00101010 DEFB 2AH ; FIXEAZA 7 BITI/CARACTER LA
; TRANSMISIE, VALIDEAZA TRANS-
; MITATORUL, RTS=1, DTR=0; SE
; INSCRIE IN WR5
4406 00000011 DEFB 03H ; INDICA WR3, SE INSCRIE IN WR0
4407 01100001 DEFB 61H ; FIXEAZA 7 BITI/CARACTER LA RE-
; CEPTIE, AUTOVALIDARI, VALIDA-
; REA RECEPTORULUI; SE INSCRIE
; IN WR3

```



4408 00000001  
4409 00010111

DEFB 01H  
DEFB 17H

; INDICA WR1, SE INSCRIE IN WR0  
; FIXEAZA INTRERUPERE PE FIE-  
; CARE CHARACTER RECEPTIONAT,  
; PARITATEA SCHIMBA VECTORUL  
; DE INTRERUPERE, VALIDEAZA IN-  
; TRERUPERI DE LA TRANSMITA-  
; TOR SI DIN EXTERIOR, STAREA  
; AFECTEAZA VECTORUL; SE IN-  
; SCRIE IN WR1



## CAPITOLUL VI

### CIRCUITUL DE INTRARE IEȘIRE PARALELĂ — Z80 PIO

#### 6.1. DESCRIEREA CIRCUITULUI DE INTRARE-IEȘIRE PARALELĂ Z80 PIO

Circuitul permite interfațarea directă a unui microsistem cu Z80 cu diferite periferice. Conține două porturi și poate fi programat pentru 4 moduri de funcționare. Interfațarea fără logică externă se poate realiza cu tastaturi, cititoare/perforatoare de bandă, imprimante, programatoare de memorii PROM/EPROM etc. Transferul de date între dispozitivul periferic și unitatea centrală are loc sub controlul dispozitivului de întreruperi, logica de întrerupere a circuitului PIO permițând utilizarea eficientă a acestuia. O altă caracteristică a circuitului PIO este posibilitatea de a întrerupe unitatea centrală la apariția unor condiții de stare specificate la dispozitivul periferic, cum ar fi anumite condiții de alarmă, reducând timpul necesar unității centrale pentru verificarea, pe rând, a stării perifericelor.

Porturile circuitului PIO sînt notate Port A și Port B. Fiecare are 8 biți de date și 2 semnale de conversație, Ready și Strobe, care controlează transferul de date. Ieșirea Ready indică perifericului că portul este gata pentru transferul de date, iar intrarea Strobe, conectată la o ieșire a perifericului, arată cînd a apărut un transfer de date.

#### Moduri de funcționare

Circuitul Z80 PIO poate funcționa în 4 moduri: ieșire de octet (Mod 0), intrare de octet (Mod 1), intrare/ieșire de octet (Mod 2) și intrare/ieșire de bit (Mod 3).

În modul 0, oricare dintre porturile A sau B poate fi programat pentru o ieșire de date, ambele porturi avînd registre de ieșire adresate individual de CPU, și în care data poate fi înscrisă în orice moment. Cînd o dată este înscrisă într-un port, o ieșire Ready activă indică dispozitivului extern că data este accesibilă la portul asociat, pentru transfer spre el. După transfer, dispozitivul extern răspunde cu un semnal activ pe intrarea Strobe, ceea ce generează o întrerupere, dacă este validată.

În modul 1, porturile A sau B iau configurația de intrare. Fiecare are un registru de intrare adresat de CPU. Cînd unitatea centrală citește o dată dintr-un port, circuitul PIO își fixează semnalul Ready, care este detectat de dispozitivul extern. Dispozitivul extern plasează în continuare data pe liniile de intrare/ieșire ale portului și dă un semnal Strobe, care determină înscrierea datelor în registrul de intrare al portului, șterge semnalul Ready, și activează semnalul de cerere de întrerupere, dacă este validat. Unitatea centrală poate citi data de intrare în orice moment, ceea ce fixează din nou Ready.

În modul 2, bidirecțional, se utilizează portul A și semnalele de întrerupere și de conversație ale ambelor porturi. Portul B trebuie programat în modul 3 și nu trebuie utilizat. Portul A este folosit atît pentru intrări cît și pentru ieșiri de date. Ieșirea este asemănătoare celei din modul 0, dar data apare la portul A doar cînd  $\overline{ASTB}=0$ . Intrarea este similară celei din modul 1, dar se utilizează semnalele de conversație și întrerupere ale portului B (dacă acesta din urmă este validat).

În modul 3 pot fi folosite ambele porturi, biții individuali ai lor fiind definiți fie ca ieșiri, fie ca intrări (cîte 8 biți pentru un port). Semnalele Ready și Strobe nu sînt utilizate. Un semnal de întrerupere poate fi generat dacă starea unei intrări nu sînt utilizate. Un semnal de întrerupere poate fi generat dacă starea unei intrări sau starea tuturor intrărilor se schimbă. Condițiile de generare a unei cereri de întrerupere sînt definite în timpul programării circuitului. Nivelul activ poate fi ales



1 sau 0 logic, iar condiția logică este fie pentru o intrare activă (SAU) fie pentru toate intrările active (SI). De exemplu, dacă portul este programat pentru intrări active pe 0 logic și funcția aleasă este SI, atunci toate intrările portului specificat trebuie să treacă la 0 logic pentru a genera o întrerupere.

Ieșirile de date sînt controlate de CPU și pot fi înscrise sau schimbate în orice moment. Unii biți individuali pot fi nefolosiți. În modul 3, semnalele de coversație nu sînt utilizate: Ready este la 0 logic, iar Strobe este dezactivat. Cînd se utilizează întreruperile de la circuitul Z80 PIO, modul de întrerupere al unității centrale Z80 CPU, trebuie să fie Modul 2.

### Structura internă

Circuitul Z80 PIO constă dintr-o interfață pentru magistrala unității centrale, o parte de logică internă de control, logică de intrare/ieșire pentru portul A, similar pentru portul B și logică de control a întreruperilor, reprezentate în fig. 6.1.

Circuitul Z80 PIO se conectează direct la unitatea centrală, fără logică externă. Logica internă de control sincronizează magistrala de date a circuitului Z80 CPU cu interfețele dispozitivului periferic (port A și port B). Porturile de intrare/ieșire (A și B) sînt identice și permit interfațarea directă la dispozitivele periferice.

### Logica unui port

Fiecare port are registre de intrare și de ieșire și logică de control a conversației. Transferurile de date între unitatea periferică și CPU utilizează registrele de intrare și de ieșire a datelor. Logica de conversație asociată fiecărui port controlează transferul de date prin registrul de intrare sau de ieșire. Registrul de control al modului (2 biți) selectează unul dintre cele 4 moduri de funcționare.

Modul de control (modul 3) utilizează celelalte registre (figura 6.2).

Registrul de control pentru intrare/ieșire specifică biții portului care sînt ieșiri și validează acești biți, ceilalți fiind intrări. Registrul de mascare și registrul de control al mascării controlează condițiile de întrerupere în modul 3. Registrul de mascare specifică biții activi ai portului și biții mascați sau inactivi.

Registrul de control al mascării specifică 2 condiții: starea activă a biților de intrare (0 sau 1) și dacă un semnal de întrerupere este generat cînd oricare bit de intrare nemascat este activ (condiția SAU) sau dacă este generat cînd toți biții de intrare nemascați sînt activi (condiție SI).

### Logica de control a întreruperilor

Logica de control a întreruperilor deține tot protocolul întreruperilor spre unitatea centrală pentru structuri de întrerupere prioritară suprapuse. Poziția fizică a unui dispozitiv într-un lanț de priorități determină prioritatea lui. Două linii (IEI

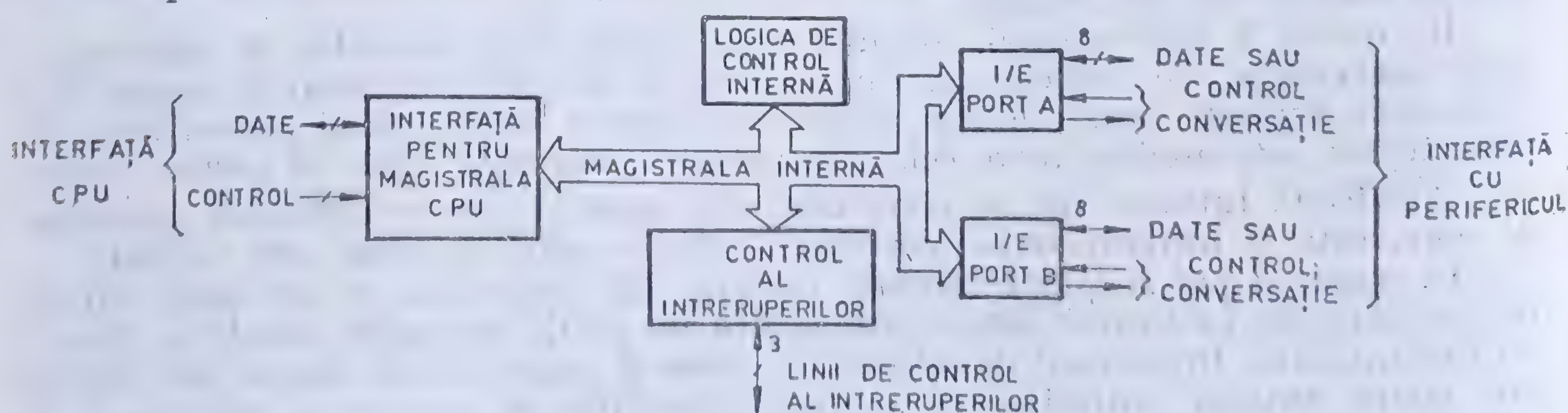


Fig. 6.1 Structura internă a circuitului Z80 PIO.



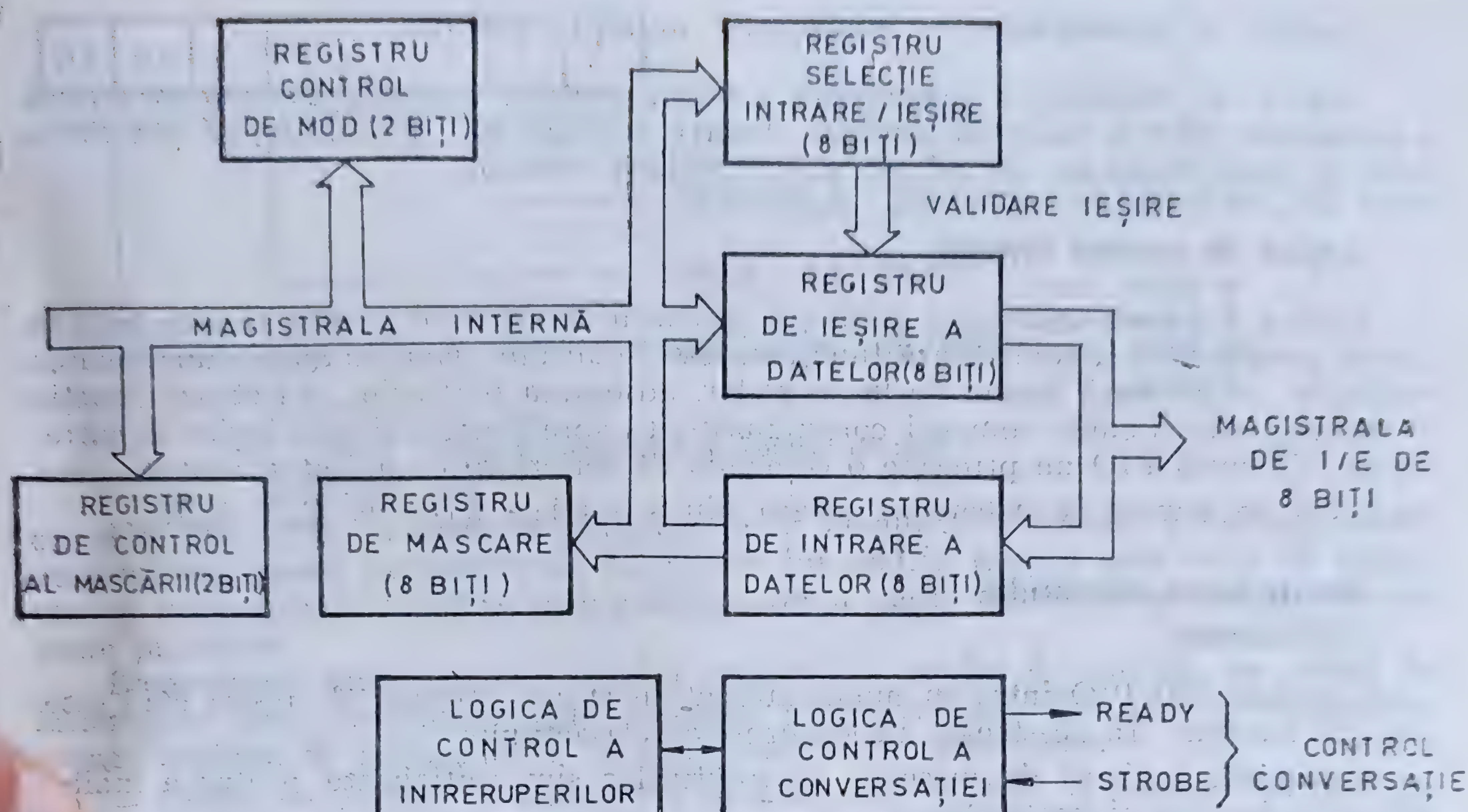


Fig. 6.2 Diagrama bloc a unui port de intrare/ieșire.

și IEO) apar la fiecare circuit PIO pentru a forma lanțul de priorități. Dispozitivul cel mai apropiat de CPU are cea mai mare prioritate. În cadrul unui circuit PIO, întreruperile portului A au o prioritate mai mare decât cele ale portului B.

În modurile de intrare de octet, ieșire de octet sau bidirecțional, o cerere de întrerupere se poate genera oricând perifericul cere transferul unui nou octet. În modul de control de bit, întreruperea poate fi generată când starea perifericului este identică cu o valoare programată. Circuitul PIO permite controlul complet al întreruperilor suprapuse. Astfel dispozitivele cu prioritate mai mică nu pot întrerupe pe cele cu prioritate mai mare, ale căror subrutine de întrerupere nu au fost terminate de unitatea centrală. Cele cu prioritate mai mare pot însă întrerupe servirea dispozitivelor mai puțin prioritare.

Dacă CPU (aflat în modul 2 de întrerupere) acceptă o întrerupere, dispozitivul care a cerut întreruperea trebuie să furnizeze unității centrale un vector de întrerupere. Acest vector indică o locație de memorie unde se află adresa rutinei de servire a întreruperii. Cei 8 biți furnizați de dispozitivul care a cerut întreruperea formează cei 8 biți mai puțin semnificativi ai indicatorului, în timp ce registrul I din CPU asigură cei 8 biți mai semnificativi.

Fiecare port (A și B) are un vector de întrerupere independent. Cel mai puțin semnificativ bit al vectorului este fixat în mod automat la 0 în interiorul circuitului PIO, pentru că indicatorul trebuie să identifice două locații adiacente de memorie pentru o adresă completă de 16 biți.

Spre deosebire de alte periferice din sistemul Z80, circuitul PIO nu acceptă întreruperi imediat după programare, ci așteaptă până când  $\overline{MI}=0$  (de exemplu în timpul aducerii unui cod de operație). Această condiție nu este importantă într-un sistem Z80, dar poate fi, dacă se utilizează alt tip de unitate centrală.

Circuitul PIO decodifică instrucțiunea de revenire din întrerupere RETI direct de pe magistrala de date a unității centrale, astfel încât fiecare circuit PIO din sistem „știe” în orice moment dacă este deservit de unitatea centrală printr-o rutină de tratare a întreruperii, nefiind astfel necesară nici o comunicare în plus cu unitatea centrală.



## Logica de intrare/ieșire a magistralei unității centrale

Logica de interfață a magistralei unității centrale permite conectarea directă a circuitului PIO la unitatea centrală. Pentru sisteme mai dezvoltate, se pot introduce și decodificatoare de adrese și/sau registre tampon.

## Logica de control internă

Logica de control internă primește cuvîntul de control pentru fiecare port în timpul programării și controlează funcționarea circuitului: sincronizează funcționarea porturilor, controlează modul de lucru al lor, adresarea porturilor, selectează funcția de intrare/ieșire și emite comenzi corespunzătoare spre porturi și spre logica de întrerupere. Circuitul PIO nu primește o comandă de scriere de la unitatea centrală, dar un astfel de semnal se generează intern din semnalele  $\overline{RD}$ ,  $\overline{CE}$ ,  $C/\overline{D}$ ,  $\overline{IORQ}$ .

## Programarea circuitului

### Inițializarea

Circuitul Z80 PIO intră în mod automat în starea inițială (de reset) cînd este pus sub tensiune. În acest caz, au loc următoarele acțiuni:

1. Ambele registre de mascare ale porturilor sînt inițializate pentru a inhiba toți biții de date ai porturilor.
2. Liniile de date ale magistrelor porturilor trec în starea de impedanță ridicată și semnalele de conversație Ready sînt inactive (la 0 logic); modul 1 este selectat în mod automat.
3. Registrele vectorilor de adresă nu sînt inițializate.
4. Ambele bistabile de validare a întreruperilor din port sînt inițializate.
5. Ambele registre de ieșire ale porturilor sînt inițializate.

În plus, față de inițializarea automată la punerea sub tensiune, circuitul PIO poate fi inițializat aplicînd un semnal  $\overline{M1}$  (figura 6.3) în absența unui semnal  $\overline{RD}$  sau  $\overline{IORQ}$ , rezultatul fiind inițializarea circuitului imediat după ce  $\overline{M1}$  devine inactiv. Scopul acestui mod de inițializare este de a permite unei singure porți externe de a genera un semnal de RESET fără o secvență de întrerupere a alimentării. Este indicat să se prevadă inițializarea circuitelor PIO în acest mod.

Este posibilă de asemenea și o inițializare prin program a circuitului, dar utilizarea acestei metode în sistemele care se pun în funcțiune poate să nu fie eficientă din cauza erorilor hard care mai pot exista.

După ce intră în starea inițială, circuitul PIO rămîne în această stare pînă la primirea unui cuvînt de control de la unitatea centrală.

## Stabilirea modului de funcționare

Programarea unui port în modul 0, 1 sau 2 (intrare de octet, ieșire de octet sau intrare/ieșire de octet) necesită două cuvinte pentru fiecare port: un cuvînt de control de mod, care selectează modul de funcționare al portului și care poate fi înscris în orice moment (figura 6.4) și un vector de întrerupere care trebuie furnizat de circuitul PIO care a cerut o întrerupere, dacă aceasta a fost acceptată (figura 6.5);

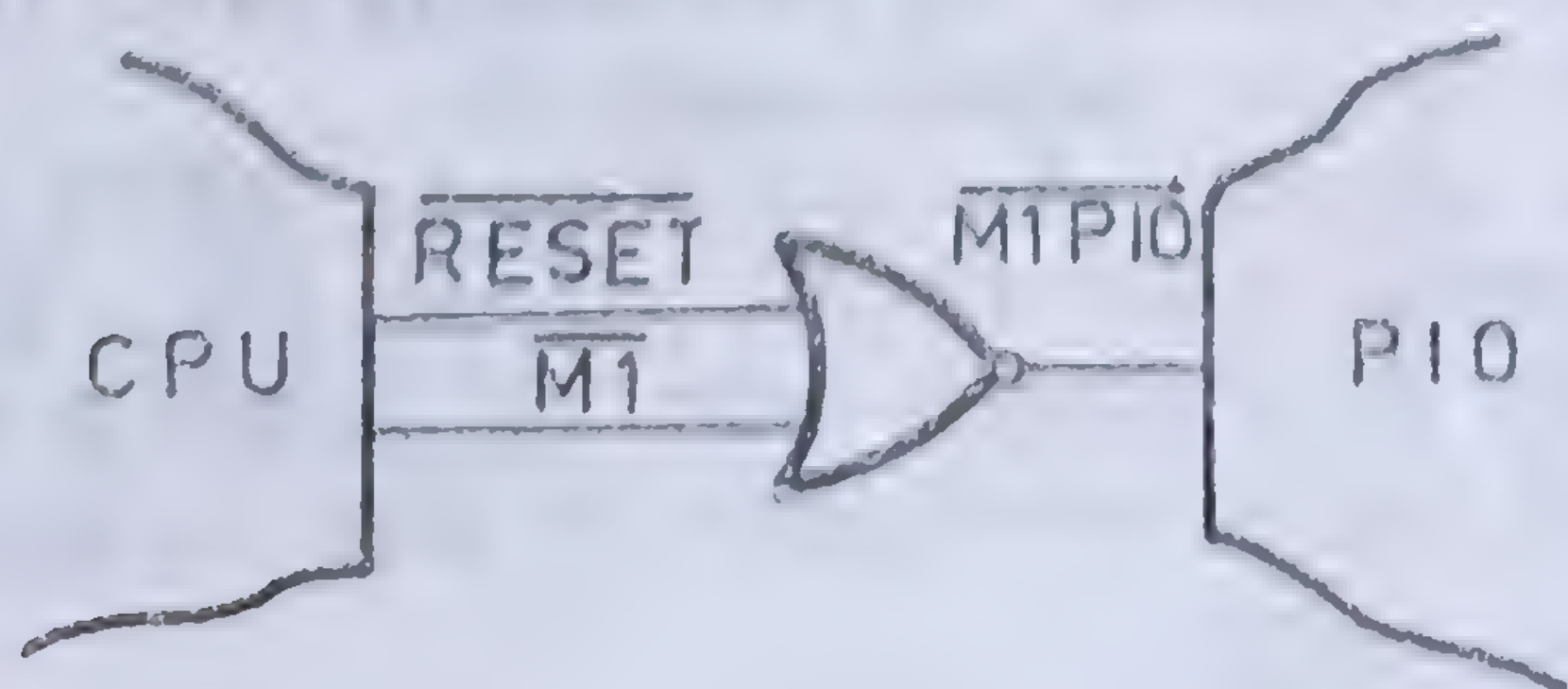
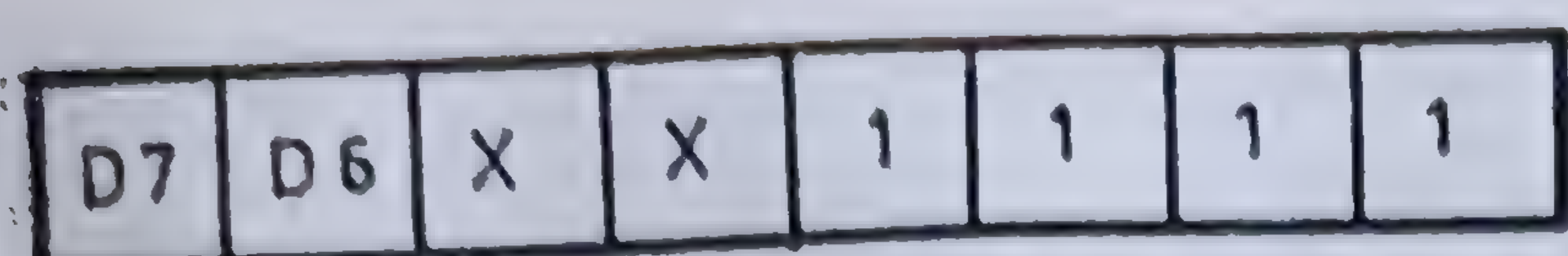


Fig. 6.3 Inițializarea circuitului Z80 PIO.





IDENTIFICĂ CUVINTUL DE CONTROL DE MOD

ORICE VALOARE

SELECȚIE DE MOD:

00	MOD 0
01	MOD 1
10	MOD 2
11	MOD 3

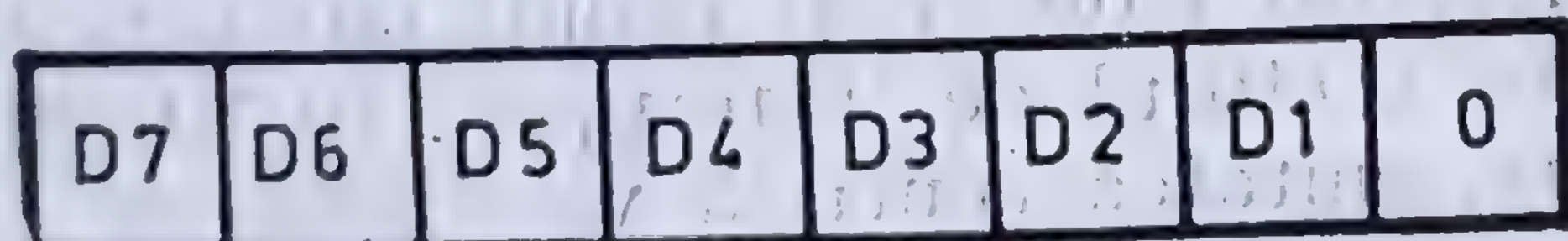
Fig. 6.4 Cuvîntul de control de mod.

circuitul Z80 CPU va trata această cerere în modul 2 de tratare a întreruperilor; vectorul este plasat pe magistrala de date a lui Z80 în timpul unui ciclu de acceptare a întreruperii, de către dispozitivul care a cerut întreruperea, avînd cea mai mare prioritate.

Programarea unui port în modul 3 necesită un cuvînt de control, un vector de întrerupere (dacă întreruperile sînt validate) și trei cuvinte adiționale (caracteristice numai modului de intrare/ieșire de bit), descrise în continuare:

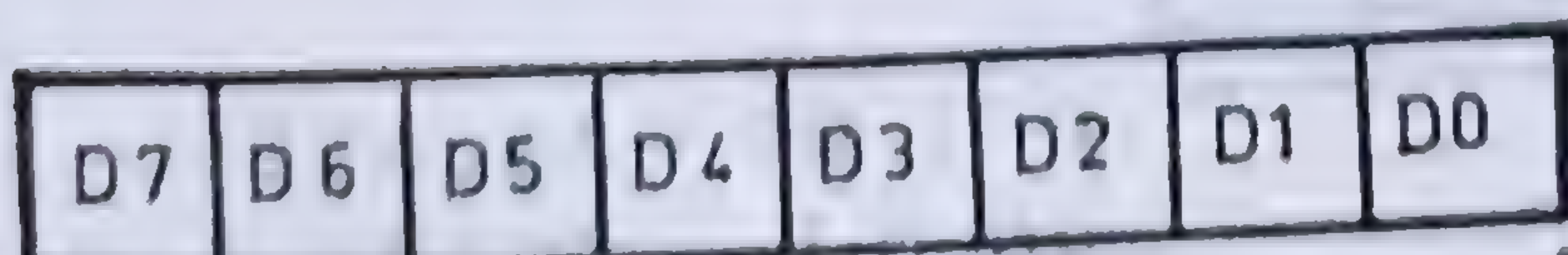
— cuvînt de control al registrului de I/E, care trebuie să urmeze cuvîntul de control de mod și fixează registrul de control al intrării/ieșirii, care, determină la rîndul lui liniile portului ce vor fi intrări și pe cele care vor fi ieșiri (figura 6.6).

— cuvînt de control al întreruperilor: în modul 3, conversația nu este utilizată; întreruperile sînt generate ca o funcție logică de nivelele semnalelor de intrare; cuvîntul de control al întreruperilor fixează condițiile logice și nivelele logice necesare pentru generarea unei întreruperi; sînt posibile două condiții (funcții) logice: SI (dacă toți biții de intrare sînt la nivelul activ, se generează o cerere de întrerupere) și SAU (o cerere de întrerupere este generată, dacă oricare dintre biții de intrare trece la nivelul activ); bitul D6 fixează funcția logică (figura 6.7); nivelul activ al biților de intrare poate fi 1 (High) sau 0 (Low) logic, și este fixat de bitul D5; de notat faptul că portul nu este validat pînă cînd validarea întreruperii nu este urmată de un semnal  $\overline{M1}$  activ:



IDENTIFICĂ VECTORUL DE INTRERUPERE  
VECTOR DE INTRERUPERE  
FIXAT DE UTILIZATOR

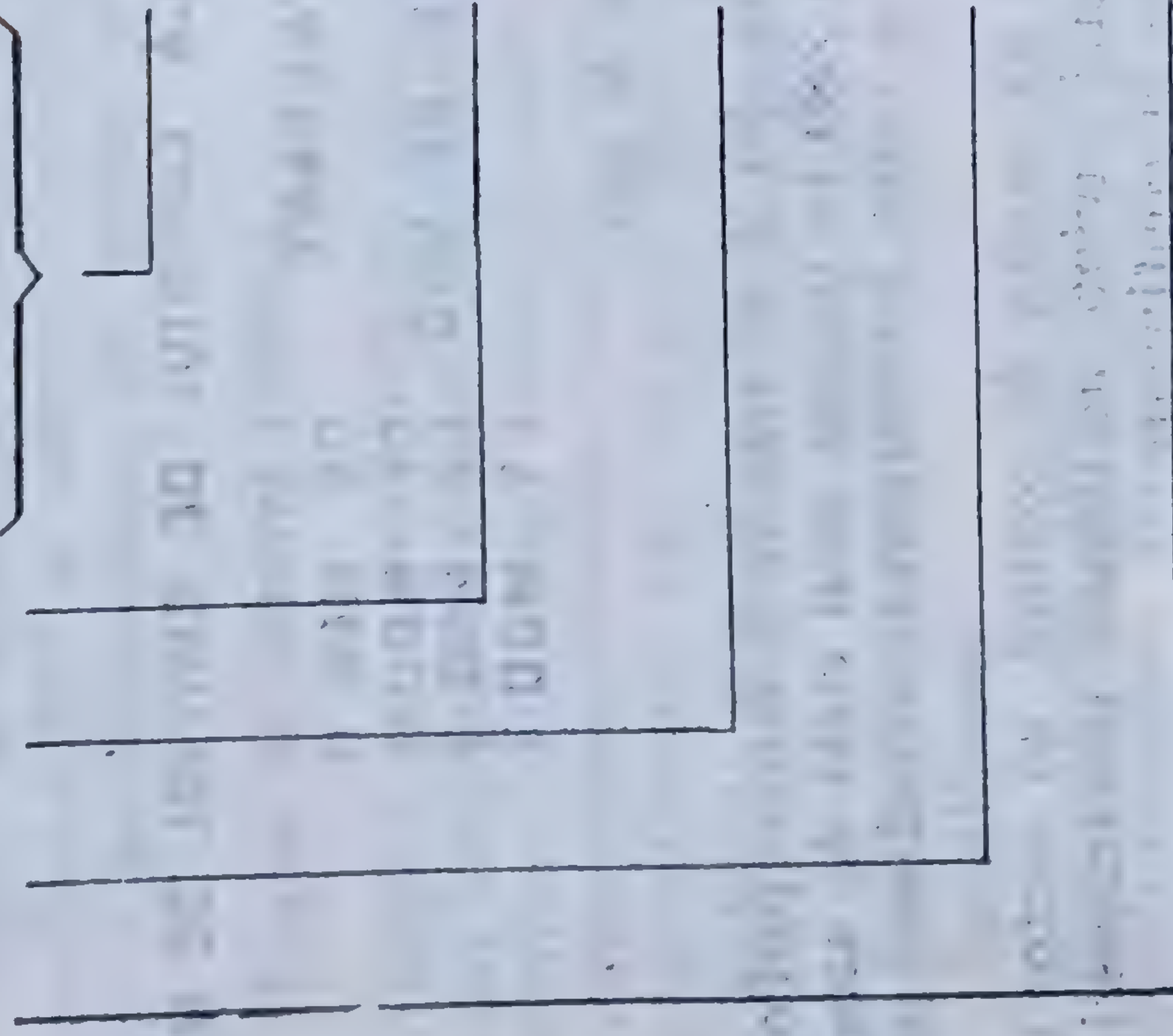
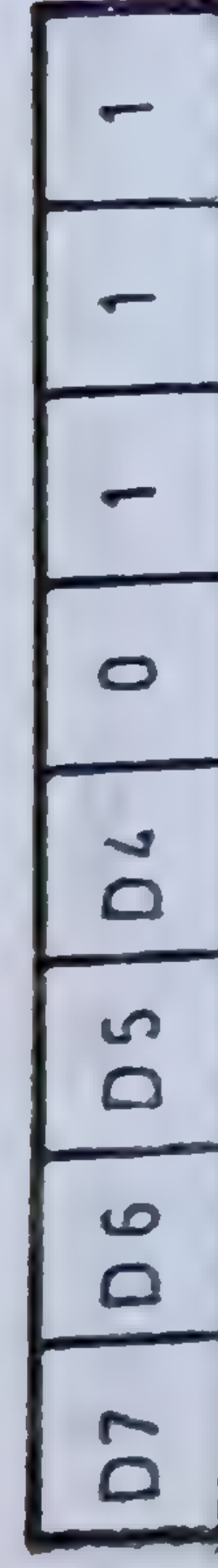
Fig. 6.5 Cuvîntul vector de întrerupere.



0 FIXEAZĂ IEȘIRILE ȘI  
1 FIXEAZĂ INTRĂRILE

Fig. 6.6 Cuvîntul de control al registrului de I/E.





IDENTIFICĂ CUVÎNTUL DE  
CONTROL AL  
INTRERUPERILOR

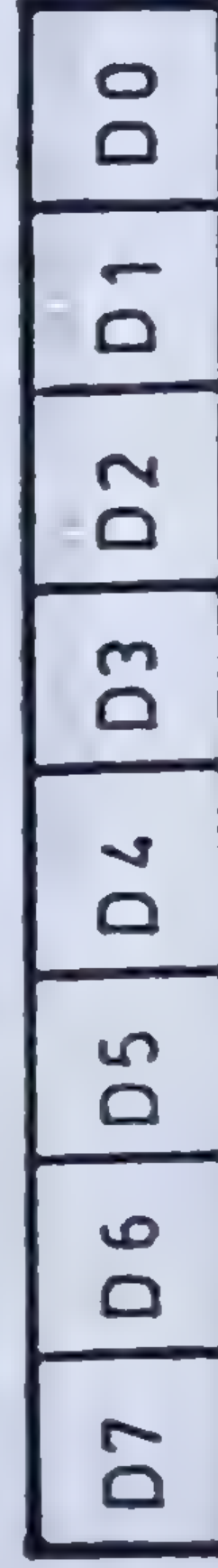
- D4=0 NU URMEAZĂ CUVÎNTUL DE  
CONTROL AL MASCĂRII
- D4=1 URMEAZĂ CUVÎNTUL DE  
CONTROL AL MASCĂRII
- D5=0 NIVELUL ACTIV ESTE 0 LOGIC
- D5=1 NIVELUL ACTIV ESTE 1 LOGIC
- D6=0 INTRERUPE PE FUNCȚIA SAU
- D6=1 INTRERUPE PE FUNCȚIA ȘI
- D7=0 INTRERUPERILE NEVALIDATE
- D7=1 INTRERUPERILE VALIDATE

Fig. 6.7 Cuvîntul de control al intreruperilor.

— cuvînt de control al mascării : fixează registrul de control al mascării, permi-  
tînd mascarea oricăror biți neutilizați; dacă există astfel de biți, atunci trebuie ca  
D4=1 în cuvîntul de control al intreruperilor; în această situație, următorul cuvînt  
înscriș în port trebuie să fie un cuvînt de control al mascării (figura 6.8).

Pentru dezactivarea intreruperilor, cuvîntul de dezactivare poate fi utilizat pentru  
a valida sau a nu valida o intrerupere de la un port. Se poate utiliza fără a schimba  
restul cuvîntului de control al intreruperilor (figura 6.9). Se stabilește în acest mod  
conținutul bistabilului de validare a intreruperilor, dintr-un port.

Dacă apare o cerere asincronă de intrerupere în timp ce procesorul înscrie cuvî-  
tul de dezactivare a intreruperilor în PIO (03<sub>H</sub>), poate să apară o problemă de  
sistem. Dacă intreruperile sînt validate în procesor, acesta va accepta intreruperea  
cerută de PIO. Totuși în acest timp, circuitul PIO va fi primit cuvîntul de dezac-  
tivare a intreruperilor și nu va trimite vectorul de intrerupere în timpul ciclului  
de recuncaștere a intreruperilor. Ca urmare, unitatea centrală va prelua de pe magis-  
trala de date un vector eronat. Soluția pentru evitarea acestei erori este să se dezac-  
tiveze intreruperile în unitatea centrală cu o instrucțiune DI chiar înainte de dezac-  
tivarea circuitului PIO și să se valideze din nou intreruperile cu o instrucțiune EI  
după aceea. Aceasta face ca unitatea centrală să ignore cererile de intrerupere de la  
circuitul PIO în timpul dezactivării lui.

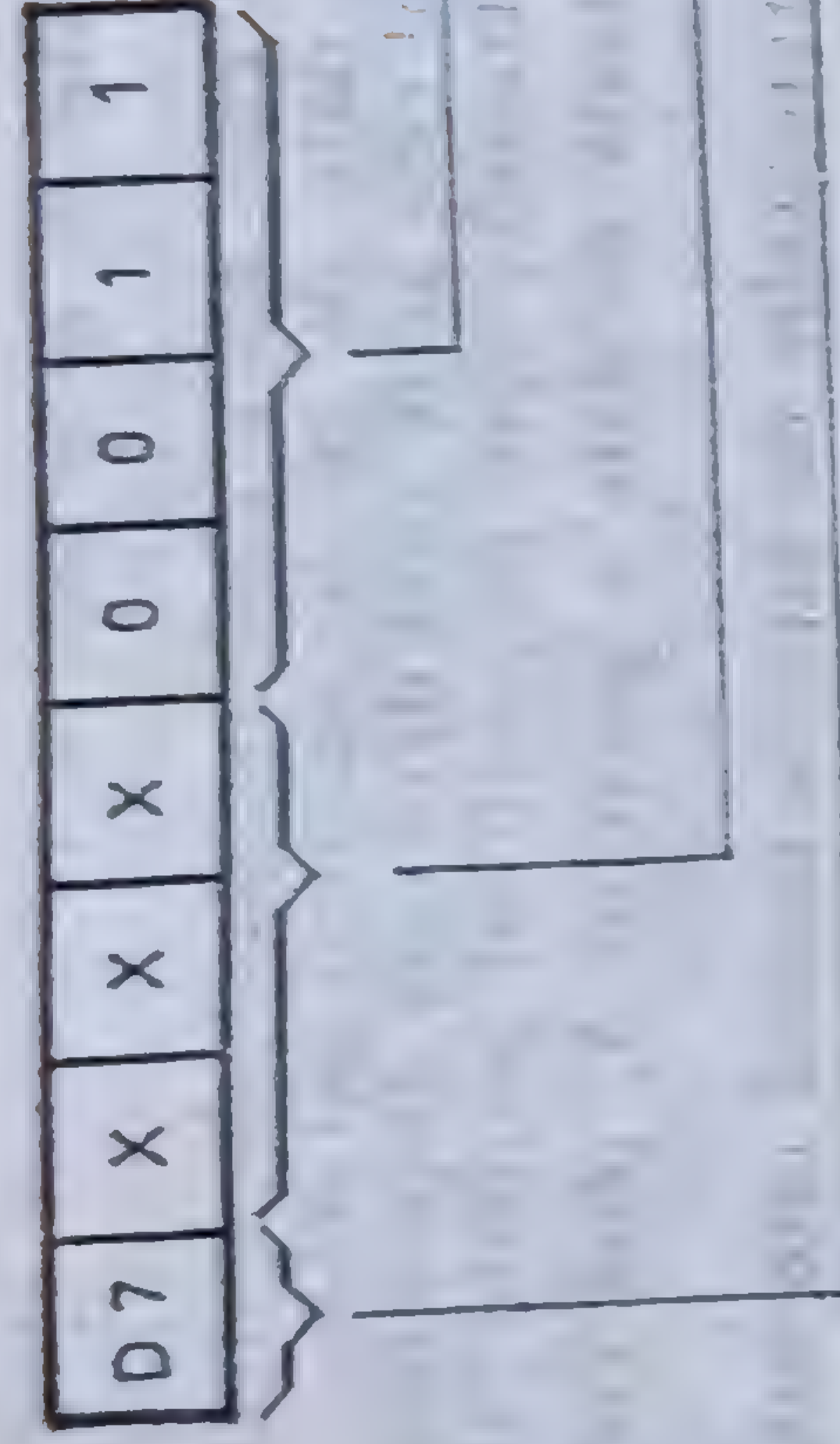


BIȚI DE MASCARE;  
UN BIT ESTE UTILIZAT  
PENTRU GENERAREA  
INTRERUPERILOR

DACĂ ESTE DEFINIT CA INTRARE ȘI BITUL DE  
MASCARE CORESPUNZĂTOR ESTE 0

Fig. 6.8 Cuvîntul de control al mascării.





D7=0 NU VALIDEAZĂ INTRERUPERILE

D7=1 VALIDEAZĂ INTRERUPERILE

Fig. 6.9 Cuvintul de dezactivare a intreruperilor.

Secvența de program care realizează aceste acțiuni este:

LD A,03H

DI

- ; CUVINT DEZACTIVARE INTRERUPERI IN PIO
- ; DEZACTIVEAZA INTRERUPERI LA CPU PE LINIA INT
- OUT (PIO), A
- EI
- ; DEZACTIVEAZA INTRERUPERI IN PIO
- ; ACTIVEAZA INTRERUPERI LA CPU PE LINIA INT

#### Conexiunile circuitului Z80 PIO

Funcțiile logice ale circuitului Z80 PIO sînt reprezentate în figura 6.10. Semnalele

de intrare sau ieșire sînt

descrie în continuare:

A0-A7 — magistrala

bidirecțională, cu

trei stări a portului

A; transferă date,

informații de stare

sau de control între

portul A al circu-

itului PIO și un dis-

pozitiv periferic; A0

este cel mai puțin

semnificativ bit.

ARDY — ieșire Ready

a portului A; semni-

ficația acestui sem-

nal depinde de mo-

dul de funcționa-

re selectat pentru

portul A:

— modul de ieșire;

semnalul devine ac-

tiv pentru a indica

faptul că registrul

de ieșire al portu-

lui A a fost încăr-

cat și că magistrala

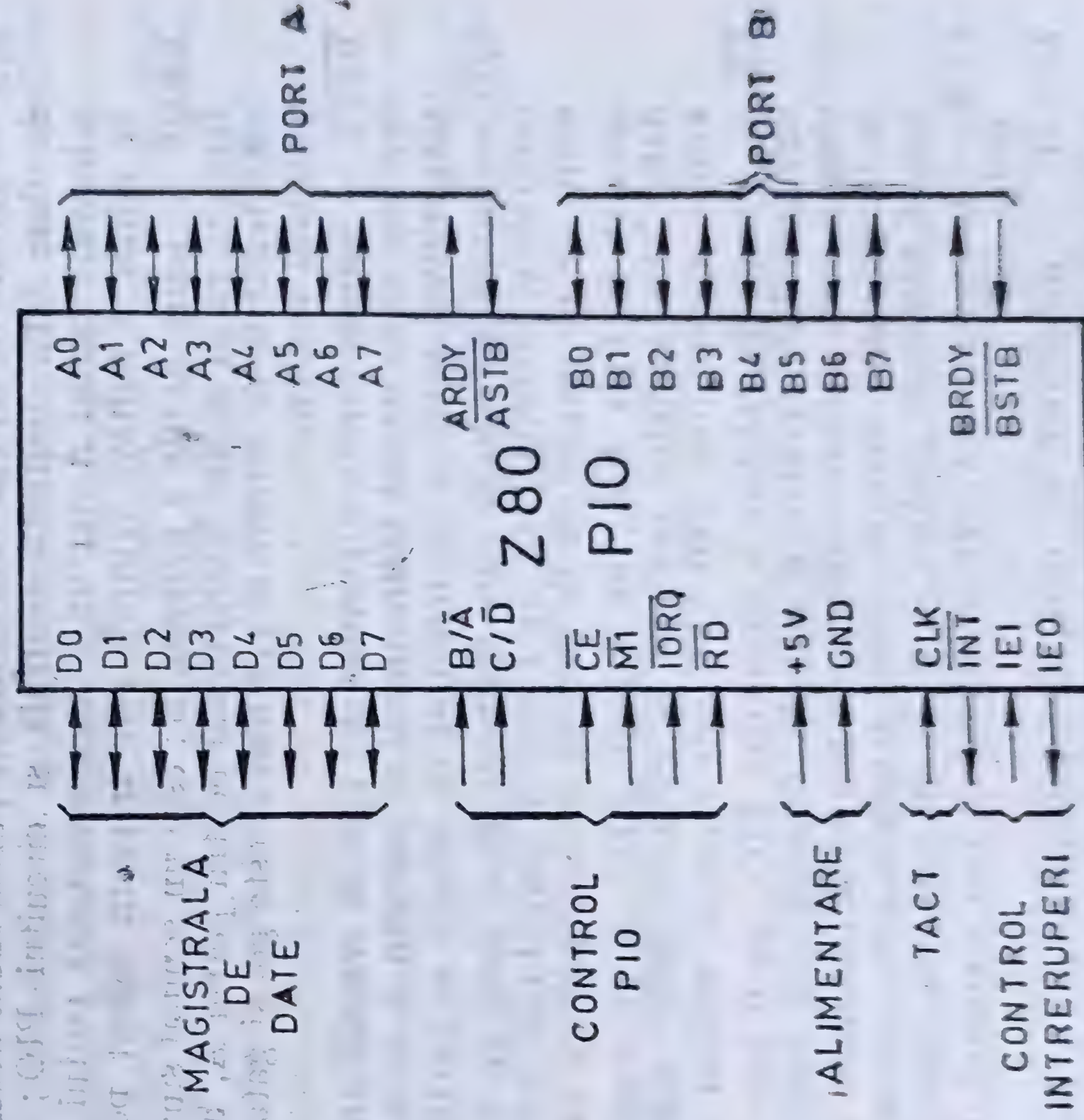


Fig. 6.10 Funcțiile logice ale circuitului Z80 PIO.



de date a perifericului este stabilă și gata pentru transferul spre dispozitivul periferic.

- modul de intrare: semnalul este activ când registrul de intrare al portului A este gol și gata să accepte date de la dispozitivul periferic.

- modul bidirecțional: semnalul este activ când data este disponibilă în registrul de ieșire al portului A, pentru transfer spre dispozitivul periferic; în acest mod, data nu este plasată pe magistrala de date a portului A, pînă când  $\overline{ASTB}$  nu este activ.

- mod de control (mod de intrare/ieșire de bit): semnalul este inhibat și adus la starea 0 logic.

**$\overline{ASTB}$**  — intrare Strobe a portului A; semnificația ei depinde de modul de funcționare ales pentru portul A:

- modul de ieșire: frontul pozitiv (crescător) al semnalului, emis de periferic, anunță primirea datei furnizată de circuitul PIO, prin portul A;

- modul de intrare: semnalul este emis de periferic, pentru a încărca data de la periferic în registrul de intrare al portului A; data este încărcată în circuitul PIO când acest semnal este activ;

- modul bidirecțional: când semnalul este activ, data din registrul de ieșire al portului A este canalizată pe magistrala de date a portului A; frontul pozitiv al semnalului anunță primirea datei;

- modul de control (intrare/ieșire de bit): semnalul este inhibat intern.

**B0—B7** — magistrala bidirecțională, cu trei stări, a portului B; transferă date, informații de stare sau de control între portul B și un dispozitiv periferic; portul B poate furniza pe fiecare linie 1,5 mA la 1,5 V pentru a comanda tranzistoare tip Darlington; B0 este cel mai puțin semnificativ bit.

**$B/\overline{A}$**  — intrare de selecție care definește portul făcut accesibil în timpul unui transfer de date între unitatea centrală și circuitul PIO; un 0 logic pe această linie selectează portul A, iar un 1 logic selectează portul B; bitul A0 al magistralei de adrese a unității centrale este frecvent folosit pentru această selecție.

**BRDY** — ieșire Ready a portului B; este un semnal similar cu ARDY, cu excepția faptului că în modul bidirecțional pentru portul A, acest semnal este la 1 logic când registrul de intrare al portului A este gol și gata să accepte date de la dispozitivul periferic.

**$\overline{BSTB}$**  — intrare Strobe a portului B; este un semnal similar cu  $\overline{ASTB}$ , cu excepția faptului că în modul bidirecțional al portului A acest semnal încarcă data de la dispozitivul periferic în registrul de intrare al portului A.

**$C/\overline{D}$**  — intrare de selecție care definește tipul datei ce se transferă între unitatea centrală și circuitul PIO; un 1 logic în timpul unei înscrieri în PIO face ca informația de pe magistrala de date să fie interpretată ca o comandă pentru portul selectat de linia  $B/\overline{A}$ ; un 0 logic arată că pe magistrala de date se transferă date între unitatea centrală și circuitul PIO; semnalul A1 al magistralei de adrese a circuitului Z80 CPU este frecvent utilizat în acest scop.

**$\overline{CE}$**  — intrare de validare a circuitului PIO; un 0 logic pe această linie validează circuitul PIO, pentru a accepta comenzi sau date de la unitatea centrală în timpul unui ciclu de scriere, sau pentru a transmite date spre unitatea centrală în timpul unui ciclu de citire; semnalul este generat de obicei decodificat din cele 4 adrese de porturi A și B, de date sau control.

**$\overline{CLK}$**  — intrare de tact; este semnalul de tact standard, cu o singură fază, al sistemului Z80.

**D0—D7** — magistrala de date, bidirecțională, cu trei stări; este utilizată pentru a transfera toate datele și comenzile între unitatea centrală și circuitul PIO; D0 este cel mai puțin semnificativ bit.



**IEI** — intrare de validare a întreruperilor; este folosită pentru a forma un lanț de priorități la cererile de întrerupere, când se utilizează mai multe dispozitive comandate prin întreruperi; un 1 logic arată că nici un alt dispozitiv cu prioritate mai mare nu este deservit de unitatea centrală în cadrul unei rutine de întrerupere

**IEO** — ieșire de validare a întreruperilor; este al doilea semnal necesar pentru a forma lanțul de priorități la întrerupere; este 1 logic doar dacă și **IEI** este la 1 logic și dacă unitatea centrală nu deservește o întrerupere de la acest PIO; astfel, acest semnal blochează cererile de întrerupere pentru dispozitivele cu prioritate mai mică în timp ce un dispozitiv cu prioritate mai mare este deservit de unitatea centrală într-o rutină de întrerupere.

**INT** — ieșire pentru cerere de întrerupere, cu drenă în gol; când este activă, circuitul PIO cere o întrerupere de la unitatea centrală Z80.

**IORQ** — cerere de intrare/ieșire; acest semnal de intrare provine de la unitatea centrală și este utilizat în combinație cu  $\overline{B/A}$ ,  $\overline{C/D}$ ,  $\overline{CE}$  și  $\overline{RD}$ , pentru a transfera comenzi și date între unitatea centrală și circuitul PIO; când  $\overline{CE}$ ,  $\overline{RD}$  și  $\overline{IORQ}$  sînt active, portul adresat de  $\overline{B/A}$  transferă date spre unitatea centrală (operația de citire) iar când  $\overline{CE}$  și  $\overline{IORQ}$  sînt active dar  $\overline{RD}$  este inactiv, portul adresat de  $\overline{B/A}$  este înscris cu date sau informații de control de la unitatea centrală, așa cum arată semnalul  $\overline{C/D}$ ; dacă  $\overline{IORQ}$  și  $\overline{M1}$  sînt simultan active, unitatea centrală anunță acceptarea unei întreruperi; portul care a cerut întreruperea plasează în mod automat vectorul lui de întrerupere pe magistrala de date a unității centrale, dacă este dispozitivul cu cel mai mare ordin de prioritate care a cerut întreruperea.

**M1** — intrare care indică primul ciclu de mașină; provine de la CPU și este utilizat ca impuls de sincronizare pentru a controla mai multe operații interne din circuitul PIO; când semnalele  $\overline{M1}$  și  $\overline{RD}$  sînt active simultan, unitatea centrală aduce o instrucțiune din memorie; când  $\overline{M1}$  și  $\overline{IORQ}$  sînt simultan active, unitatea centrală anunță acceptarea unei întreruperi; în plus,  $\overline{M1}$  mai are două funcții în circuitul PIO: sincronizează logica de întrerupere din PIO și, când  $\overline{M1}$  apare fără un semnal activ  $\overline{RD}$  sau  $\overline{IORQ}$ , circuitul PIO este inițializat.

**RD** — intrare care indică o operație de citire; semnalul provine de la unitatea centrală; dacă  $\overline{RD}$  este activ sau dacă este în curs de efectuare o operație de intrare/ieșire,  $\overline{RD}$  este folosit cu semnalele  $\overline{B/A}$ ,  $\overline{C/D}$ ,  $\overline{CE}$  și  $\overline{IORQ}$  pentru a transfera date de la circuitul Z80 PIO spre unitatea centrală.

## Funcționarea în timp a circuitului Z80 PIO

### Modul de ieșire (Mod 0)

Ciclul de ieșire începe cu execuția de către unitatea centrală a unei instrucțiuni de ieșire. Impulsul  $WR^* = \overline{RD} + \overline{CE} + \overline{C/D} + \overline{IORQ}$  introduce data de pe magistrala CPU în registrul de ieșire al portului selectat și fixează bistabilul Ready la 1 după un front negativ al CLK, arătînd că data este disponibilă. Ready este activ pînă la frontul pozitiv al liniei Strobe, care arată că data a fost preluată de periferic. Frontul pozitiv al impulsului Strobe generează un **INT** activ, dacă bistabilul de validare a întreruperilor a fost înscris și dacă acest dispozitiv are cea mai mare prioritate (figura 6.11).



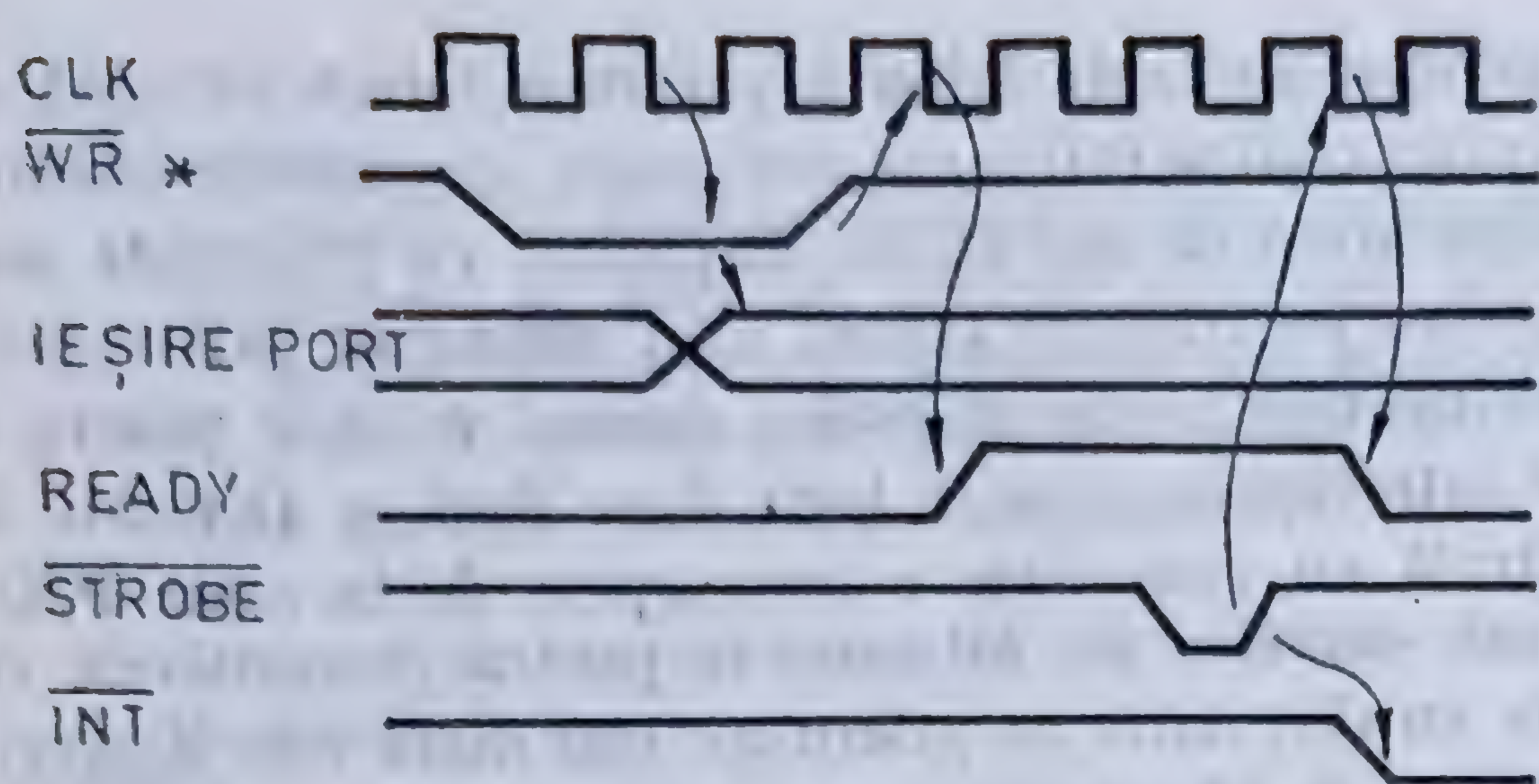


Fig. 6.11 Diagramă în timp pentru modul 0.

de intrare este plin și nu mai poate accepta date pînă cînd unitatea centrală nu efectuează o citire. După efectuarea citirii, frontul pozitiv al lui  $\overline{RD}$  fixează Ready la următorul front negativ al lui CLK, ceea ce face ca o nouă dată să poată fi încărcată în circuitul PIO. În figură semnalul  $\overline{RD}^*$  este calculat după formula  $\overline{RD}^* = \overline{RD} + \overline{CE} + \overline{C/D} + \overline{IORQ}$ .

#### Modul bidirecțional (Mod 2)

Acest mod este o combinație a modurilor 0 și 1, utilizînd toate cele 4 semnale de conversație și liniile de date ale portului A (figura 6.13). Portul B trebuie să fie programat în modul 3 și intrările lui trebuie să fie mascate. Liniile de conversație

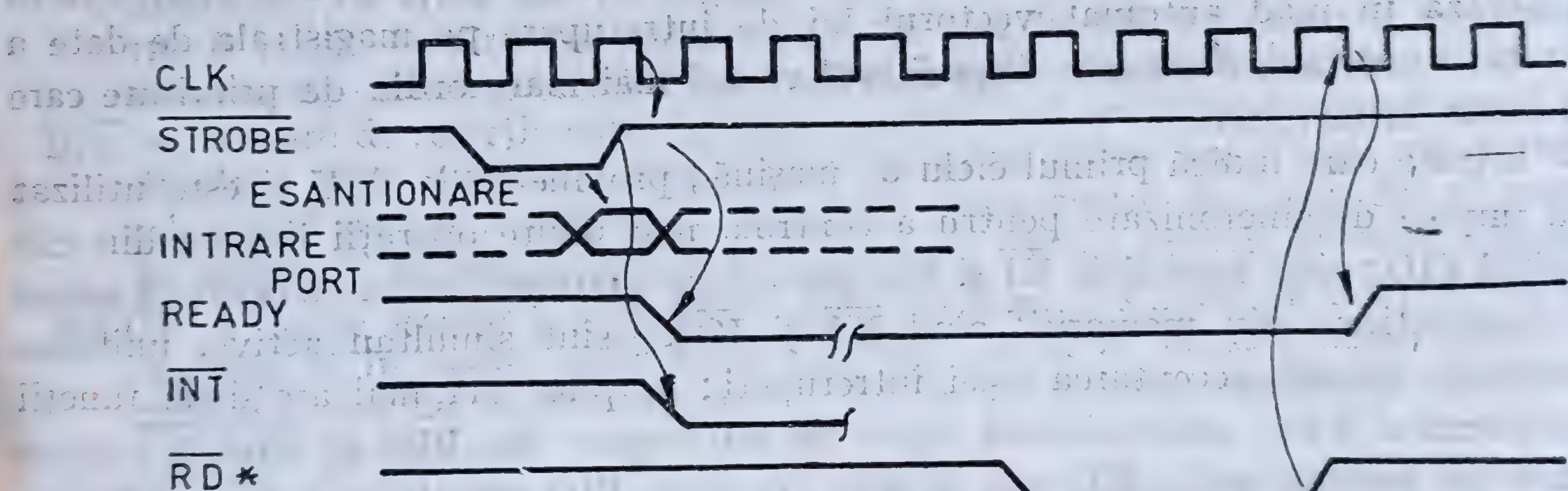


Fig. 6.12 Diagramă în timp pentru modul 1.

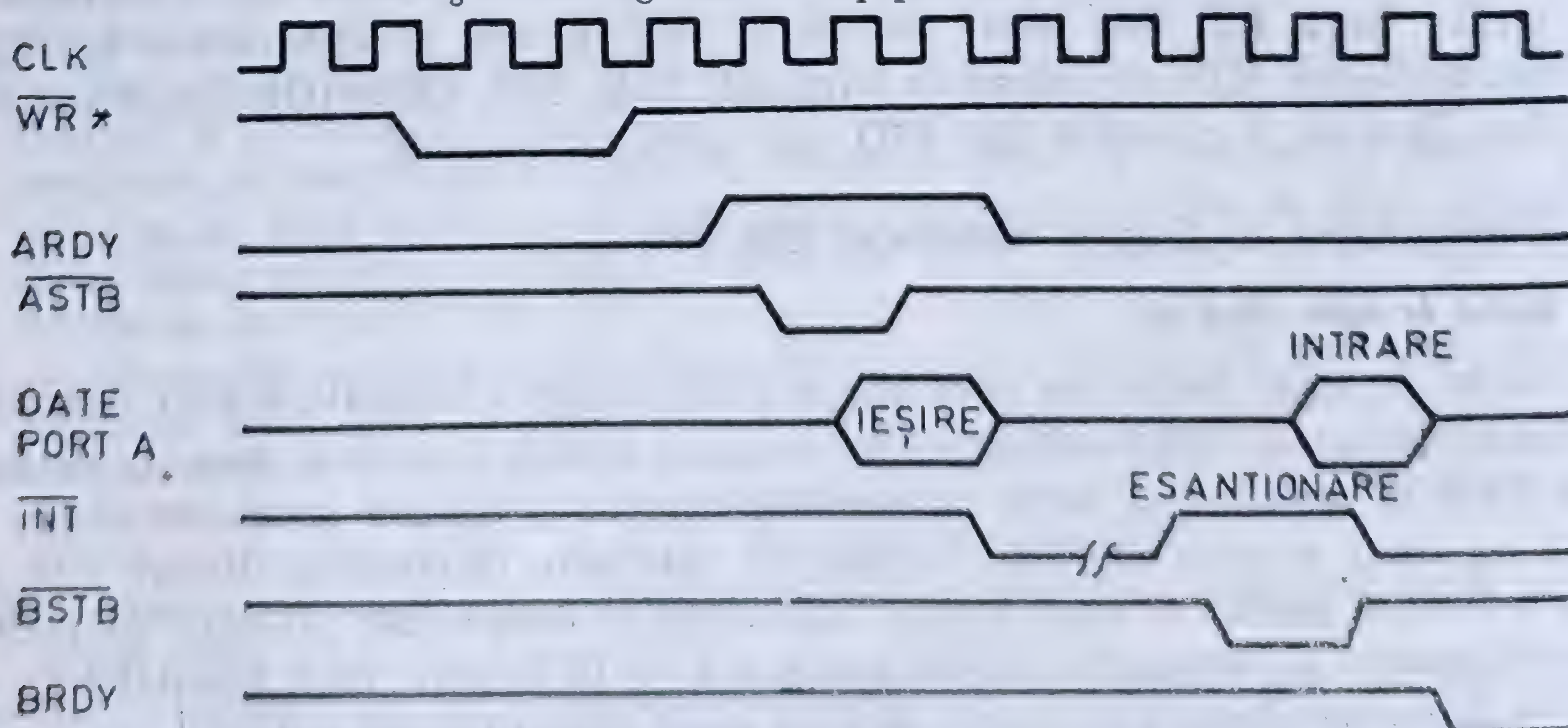


Fig. 6.13 Diagramă în timp pentru modul 2.



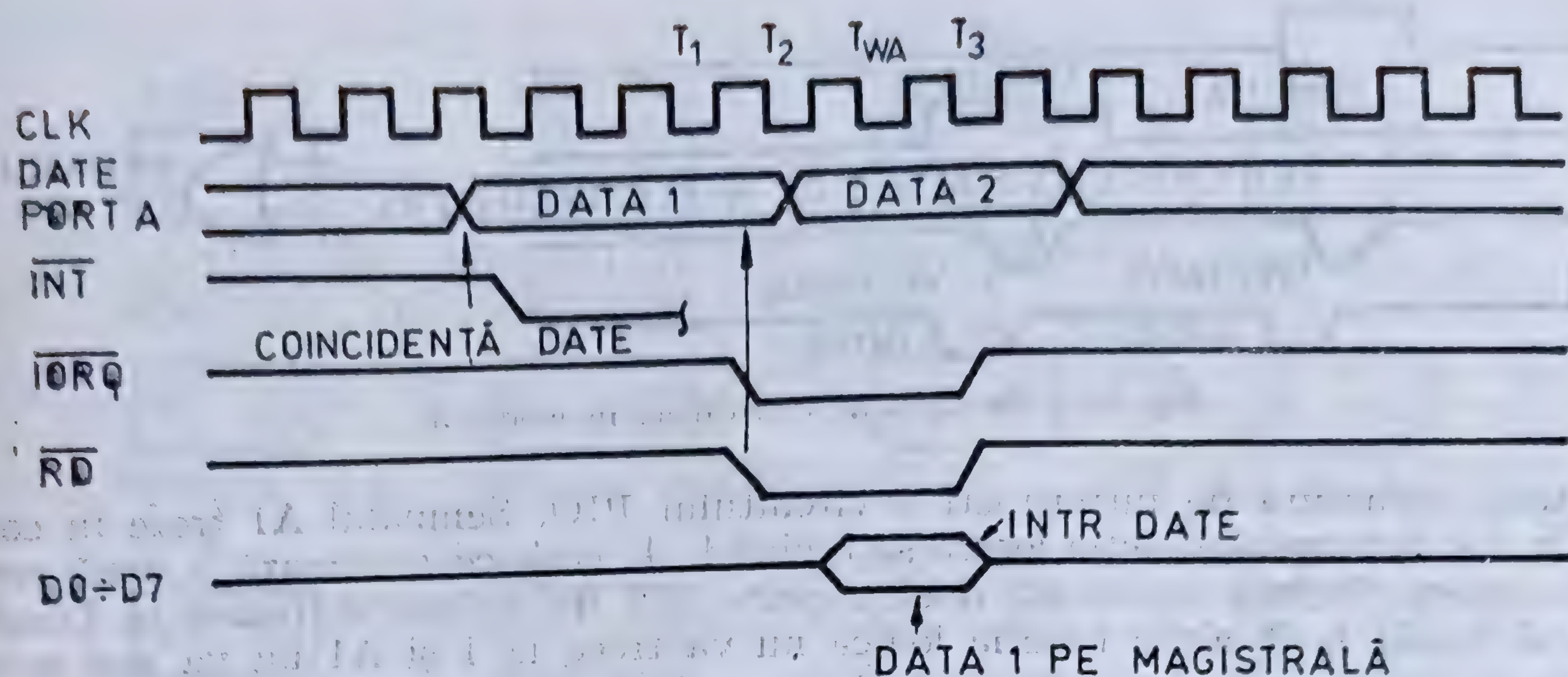


Fig. 6.14 Diagramă în timp pentru citire în modul de control.

ale portului A sînt utilizate pentru controlul ieșirii, iar liniile portului B sînt utilizate pentru controlul intrării. Dacă apare o întrerupere, vectorul portului A va fi utilizat în timpul ieșirii pe port, iar vectorul portului B, în timpul intrării pe port. Datele apar la ieșirea portului A numai cînd  $\overline{ASTB}$  este la 0. Frontul pozitiv al acestui semnal poate fi utilizat pentru a încărca data în periferic.

#### Modul de control (Mod 3)

Acest mod nu utilizează semnalele de conversație și o înscrisoare sau citire de port poate fi făcută în orice moment. La scriere, data este încărcată în registrele de ieșire, după aceeași diagramă în timp ca și în modul de ieșire (figura 6.14). La citire, data care ajunge la unitatea centrală este compusă din datele din registrul de ieșire, corespunzînd liniilor portului, care sînt ieșiri și din datele din registrul de intrare corespunzînd liniilor portului care sînt intrări. Registrul de intrare conține datele care erau prezente înainte a frontului negativ al semnalului  $\overline{RD}$ . Se poate genera o întrerupere dacă întreruperile de la port sînt validate și dacă datele de pe liniile portului satisfac ecuația logică definită de registrul mască de 8 biți și de registrul de control al mascării, de 2 biți.

Dacă portul A este programat bidirecțional, iar portul B în modul de control, portul B nu va putea emite o cerere de întrerupere și trebuie verificată periodic starea lui de către unitatea centrală. De exemplu, dacă presupunem că s-a ales condiția logică „SAU” și o linie de date nemascată a portului devine activă, se va cere o întrerupere. Dacă o a doua linie de date nemascată devine activă în același timp cu prima, nu se va cere o nouă întrerupere dacă nu a apărut o schimbare în rezultatul funcției logice aleasă pentru modul 3. De notat că semnalele portului definite ca ieșiri pot contribui la ecuația logică, dacă pozițiile lor nu sînt mascate. Dacă rezultatul funcției logice devine „1” imediat înainte sau în timpul unui semnal  $\overline{MI}$ , o întrerupere se va cere după frontul de terminare al lui  $\overline{MI}$ , cu condiția ca funcția logică să rămîină la „1” după ce  $\overline{MI}$  revine la 1 logic (figura 6.15). Toți biții, în afară de A0 și A1 sînt mascați, rezultînd o funcție logică SAU cu 2 intrări, în logica pozitivă. Treccrea lui A0 la 1 crează o întrerupere ( $\overline{INT}$  trece la 0) și unitatea centrală răspunde cu un ciclu de recunoaștere a întreruperii ( $\overline{INTACK}$ ). Circuitul PIO trimite vectorul de întrerupere spre unitatea centrală, care trece la execuția rutinei de servirea a întreruperii. Semnalul A0 devine inactiv fie singur, fie ca rezultat al acțiunilor rutinei de servire a întreruperii, determinînd funcția logică să treacă la „0”. O săgeată indică momentul în care rutina de întrerupere emite o instrucțiune RETI.



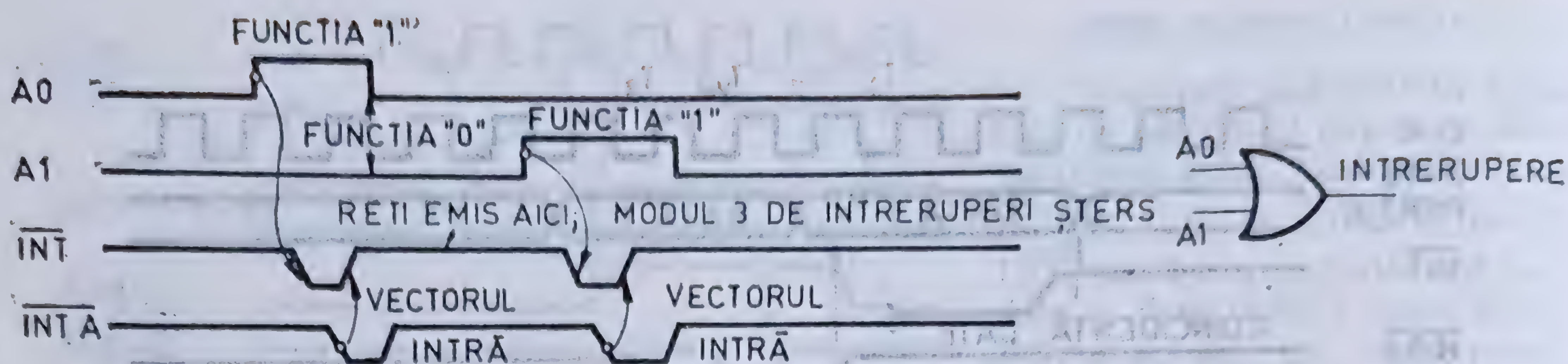


Fig. 6.15 Exemplu de întrerupere în modul 3.

care șterge structura de întreruperi a circuitului PIO. Semnalul A1 trece în continuare la 1 și determină funcția logică să revină la 1, ceea ce generează o nouă cerere de întrerupere. Trebuie remarcate două aspecte: A1 nu trebuie să treacă la 1 înainte ca A0 să treacă la 0, altfel funcția logică nu va trece la 1 și A1 nu va mai genera o cerere de întrerupere; pentru ca A1 să genereze o cerere de întrerupere, trebuie să treacă la 1 după ce instrucțiunea RETI emisă de rutina de servire a întreruperii pe A0 a șters structura internă de întreruperi a circuitului PIO; Cu alte cuvinte, dacă A1 este un impuls pozitiv care a apărut după ce A0 a trecut la 0, determinând funcția logică să fie „0”, și a dispărut înainte ca RETI să fi șters (inițializat) structura de întreruperi, nu se generează o întrerupere; funcția logică trebuie să devină 0 după recunoașterea întreruperii pe A0 și trebuie să fie sau să devină 1 după ce RETI șterge întreruperea anterioară pentru ca o altă întrerupere să apară.

În cazul programării portului A în modul 2 și portului B în modul 3, același vector de întrerupere va fi furnizat unității centrale pentru o întrerupere la portul B sau una la transferul de intrare în portul A. Se poate evita această ambiguitate, dacă starea portului B este controlată periodic (polling) și registrul de mascare al portului B este fixat pentru a inhiba toți biții. Ca urmare, nu se vor mai genera cereri de întrerupere de la portul B (în modul 3) când portul A este programat în modul 2, deoarece BSTB ar trebui să fie activ (0 logic) pentru a genera întreruperi (BSTB este în mod obișnuit la 1 logic).

### Recunoașterea unei întreruperi

În timpul perioadei active a lui  $\overline{M1}$ , controlerle periferice nu pot să schimbe starea validării întreruperilor, permițând semnalului de validare a întreruperilor Interrupt Enable să parcurgă tot lanțul de priorități (maximum 4 circuite PIO). Perifericul cu  $IEI = 1$  și  $IEO = 0$  în timpul impulsului  $\overline{INTACK}$  plasează în acest timp un vector de întrerupere programat anterior, pe magistrala de date. Semnalul IEO este menținut la 0 pînă cînd unitatea centrală execută o instrucțiune RETI în timp ce IEI este la 1. Instrucțiunea de 2 octeți RETI este decodificată intern de circuitul PIO în acest scop.

### Revenirea din întrerupere

Dacă un periferic Z80 nu este în așteptarea sau în cursul deservirii unei întreruperi, atunci  $IEI = IEO$ . Dacă este în timpul deservirii unei cereri de întrerupere (dacă a emis o cerere și a primit un semnal de recunoaștere a întreruperii), atunci  $IEO = 0$ , inhibînd cererile de întrerupere ale unor dispozitive mai puțin prioritare. Dacă a emis o cerere de întrerupere, dar nu a primit încă un semnal de recunoaștere a ei, atunci  $IEO = 0$ , dacă nu este decodificat un cod  $ED_H$  ca prim octet al unui cod de operație de 2 octeți. În acest caz,  $IEO$  trece la 1, pînă cînd următorul cod de operație este decodificat, cînd devine din nou 0. Dacă al doilea octet al codului operației a fost  $4D_H$ , s-a executat o instrucțiune RETI, de revenire din întrerupere. După decodificarea codului  $ED_H$ , doar dispozitivul periferic care a cerut întreruperea și este în curs de servire



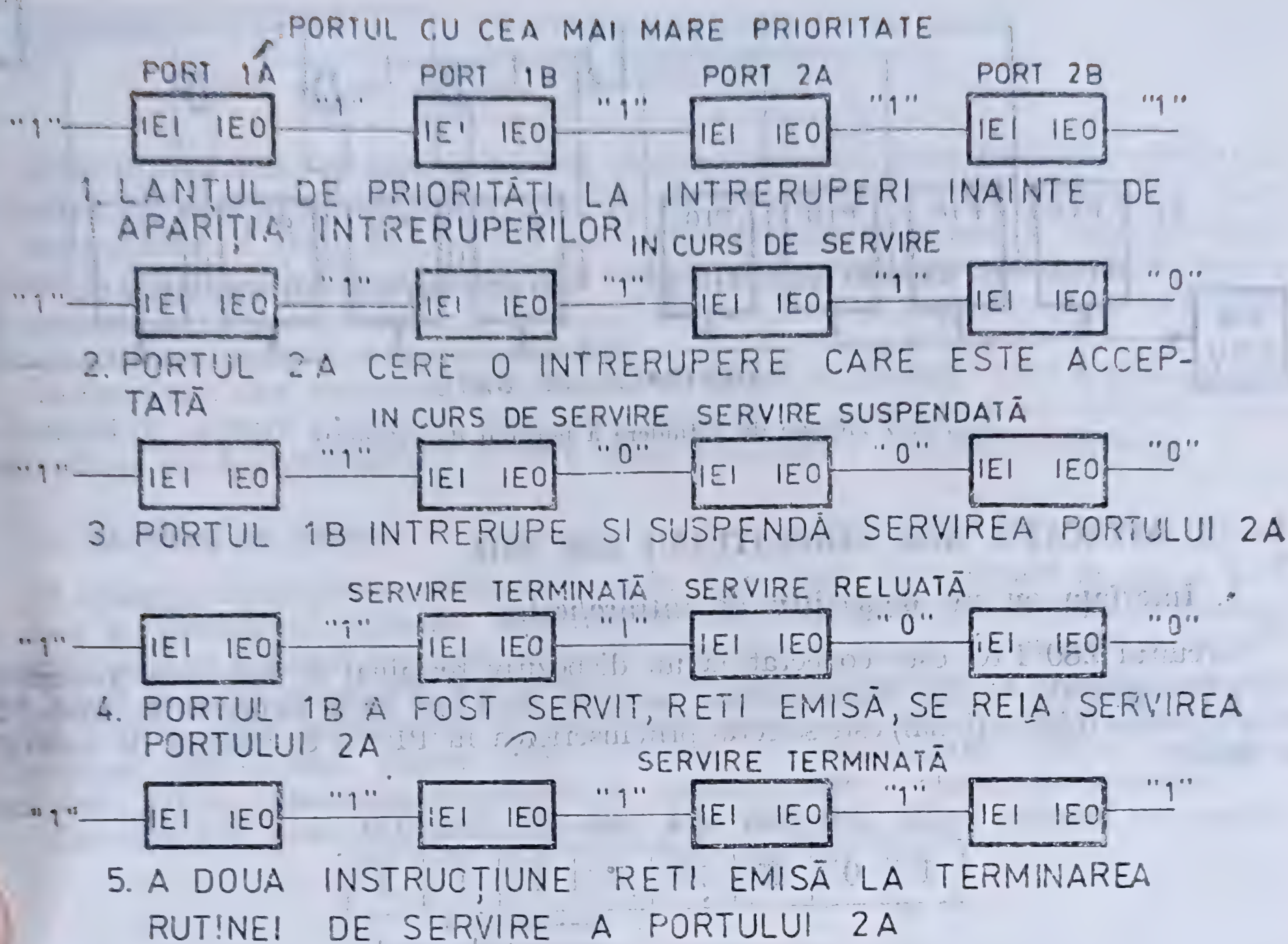


Fig. 6.16 Servirea întreruperilor într-un lanț de priorități.

are IEI la 1 logic și IEO la 0. Acesta este dispozitivul cu cea mai mare prioritate din lanțul de priorități, care a primit un semnal de recunoaștere a întreruperii. Toate celelalte periferice au  $IEI = IEO$ . Dacă următorul octet al codului operației este  $4D_H$ , acest dispozitiv periferic își anulează condiția de „întrerupere în curs de servire”.

Un exemplu tipic de întreruperi suprapuse care pot să apară într-un lanț de 4 porturi este prezentat în figura 6.16. În această secvență, portul 2A solicită o întrerupere care este acceptată. În timpul servirii acestui port, un port cu prioritate mai mare, 1B, cere o întrerupere, de asemenea acceptată. La terminarea rutinei de servire a portului 1B se execută o instrucțiune RETI pentru a anunța portului acest fapt. În continuare, este reluată și terminată rutina de servire a portului 2A.

### Extinderea lanțului de priorități

Un lanț de priorități fără logică externă conține maximum 4 circuite PIO, astfel încât starca de validare a întreruperii să se propage prin întregul lanț între începutul lui  $\overline{M1}$  și cel al lui  $\overline{IORQ}$ , în timpul unui ciclu de recunoaștere a întreruperii. Cum starea de validare a întreruperii nu se poate schimba în timpul lui  $\overline{M1}$ , vectorul de adresă furnizat unității centrale este sigur de la dispozitivul cu cea mai mare prioritate care a cerut o întrerupere.

Pentru conectarea a mai mult de 4 circuite PIO poate fi utilizată o structură de calcul în avans a semnalului de validare a întreruperilor, ca în figura 6.17, fiind astfel posibilă conectarea a aproximativ 30 de circuite în lanțul de priorități.



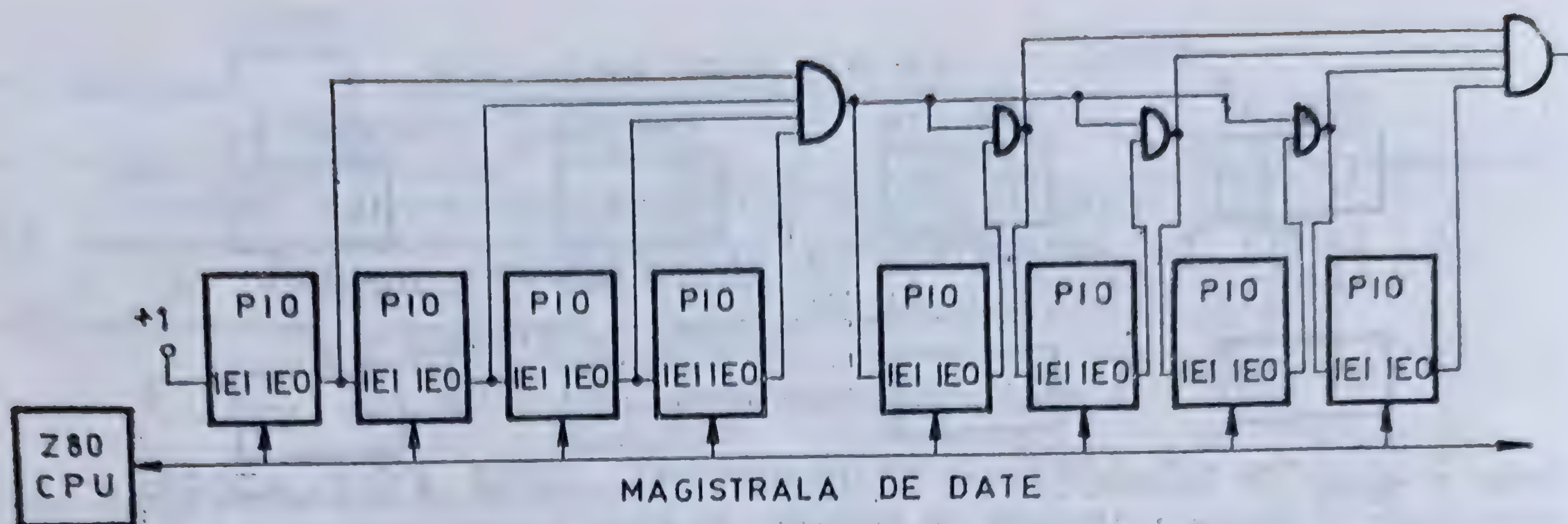


Fig. 6.17 Metode de extindere a lanțului de priorități.

## 6.2. APLICAȚII ALE CIRCUITULUI Z80 PIO

### 1. Interfața cu un dispozitiv de intrare/ieșire

Circuitul Z80 PIO este conectat la un dispozitiv terminal de I/E, care comunică printr-o magistrală de date bidirecțională, paralelă, de 8 biți, ca în figura 6.18. Modul 2 de funcționare (bidirecțional) este selectat prin înscrierea în PIO a cuvântului de control următor :

D7	D6	D5	D4	D3	D2	D1	D0
1	0	X	X	1	1	1	1

control de mod

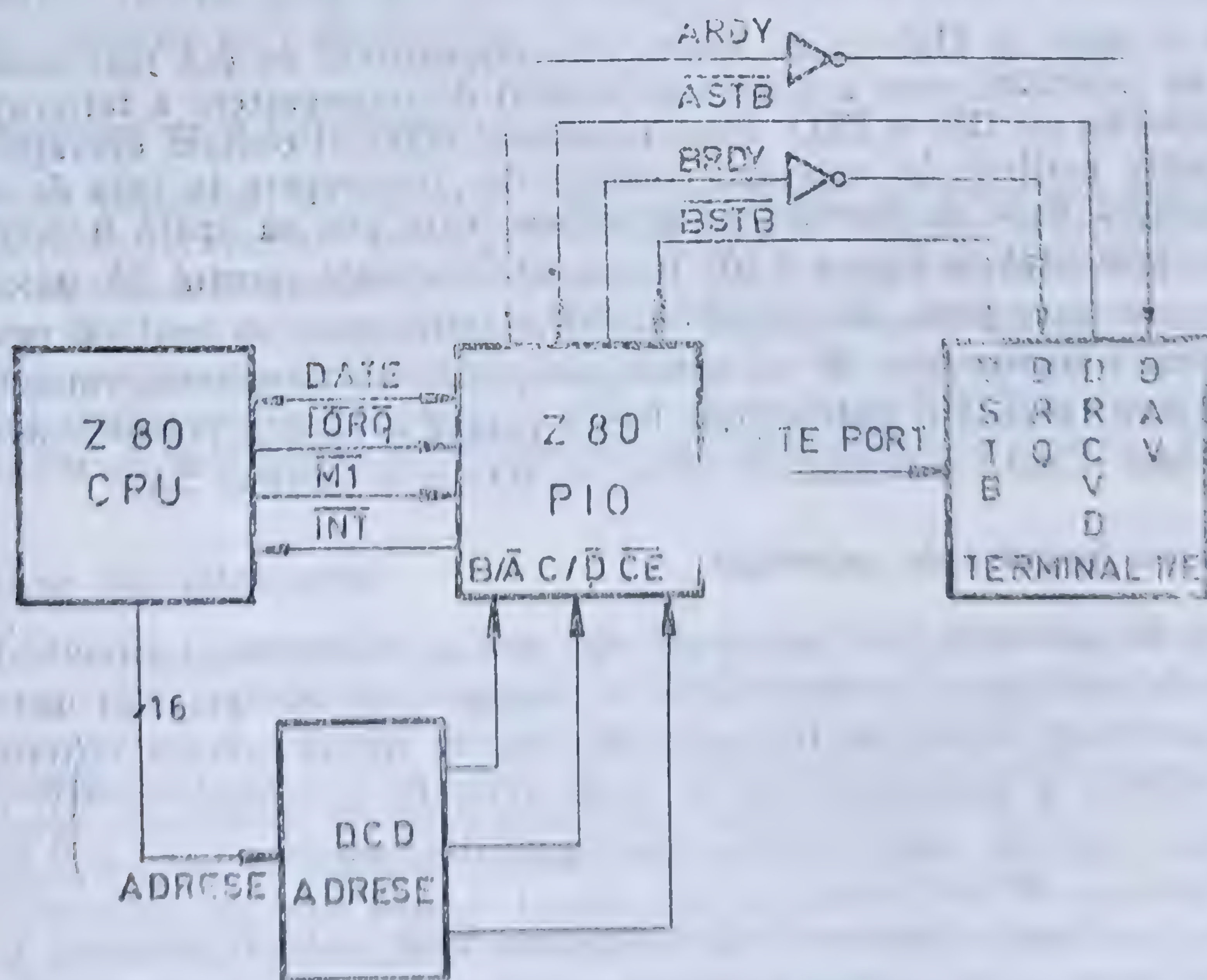


Fig. 6.18 Exemplu de interfață de intrare/ieșire.



În continuare, se înscrie în PIO vectorul de întreruperi:

V7	V6	V5	V4	V3	V2	V1	0
----	----	----	----	----	----	----	---

Întreruperile sînt validate de frontul pozitiv al lui  $\overline{M1}$ , după înscrierea cuvîntului de control de mod al întreruperilor, cu excepția cazului în care  $\overline{M1}$  definește un ciclu de recunoaștere a întreruperii. Dacă un cuvînt de mascare urmează după cuvîntul de mod al întreruperilor, întreruperile sînt validate de frontul pozitiv al primului impuls  $\overline{M1}$  urmînd înscrierea cuvîntului de mascare. Datele pot fi transferate acum între periferic și unitatea centrală, prin circuitul PIO.

În exemplul ales, semnalele terminalului de intrare/ieșire sînt: DSTB (data stroba — semnal de captare a datelor), DRQ (data request — cerere de date), DRCVD (date received — datele recepționate) și DAV (data available — datele disponibile).

## 2. Interfață de control

O aplicație tipică pentru utilizarea modului de control este ilustrată în figura 6.19. în care se presupune existența unui proces industrial al cărui control este necesar. Apariția oricăror condiții anormale de funcționare trebuie anunțată unui sistem de control bazat pe microprocesorul Z80. Cuvîntul de stare și control al procesului au formatul: D7 — test special; D6 — conectarea alimentării; D5 — alarmă la pană de alimentare; D4 — stop proces; D3 — alarmă de temperatură; D2 — conectarea încălzirii; D1 — conectarea instalației de presiune; D0 — alarmă de presiune.

Circuitul PIO poate fi folosit cu portul A în modul 3, avînd cuvîntul de control:

D7	D6	D5	D4	D3	D2	D1	D0
1	1	X	X	1	1	1	1

Cuvîntul de selecție de I/E pentru liniile A5, A3, A0 ca intrări, este:

0	0	1	0	1	0	0	1
---	---	---	---	---	---	---	---

În continuare se înscrie în PIO vectorul de întreruperi:

V7	V6	V5	V4	V3	V2	V1	0
----	----	----	----	----	----	----	---

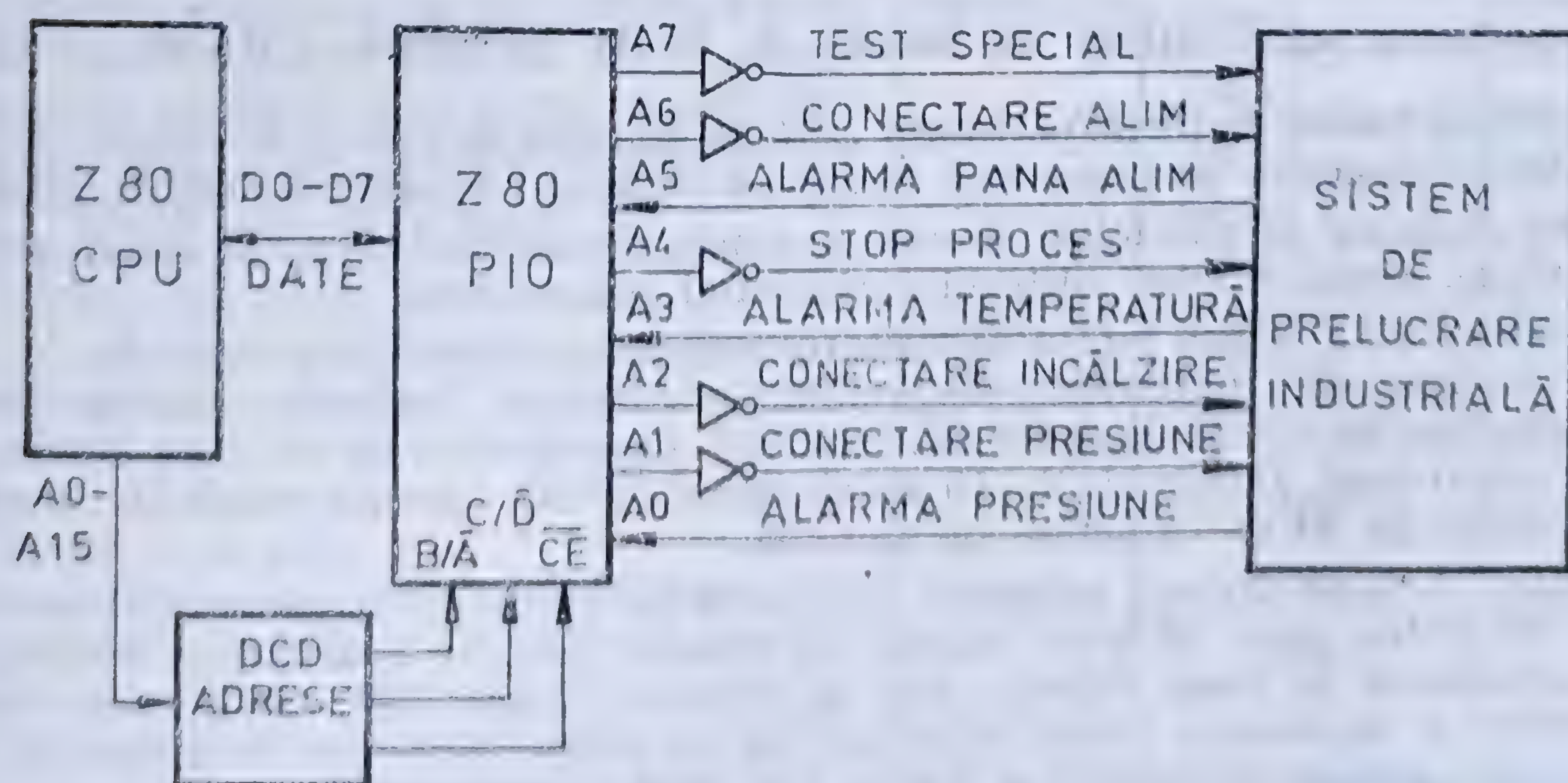


Fig. 6.19 Aplicație în modul de control.



Urmează cuvîntul de control al întreruperilor, înscris în portul A :

1	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

în care bitul  $D7 = 1$  validează întreruperile,  $D6 = 0$  fixează logica „SAU”,  $D5 = 1$  fixează nivelul activ „1”,  $D4 = 1$  arată că urmează cuvîntul mască, iar ultimii 4 biți identifică cuvîntul de control al întreruperilor.

Cuvîntul mască, care selectează A5, A3 și A0 pentru calculul funcției de întrerupere este:

1	1	0	1	0	1	1	0
---	---	---	---	---	---	---	---

În continuare, dacă un senzor dă valoarea „1” pe liniile A5, A3 sau A0, se va genera o cerere de întrerupere. Cuvîntul mască poate selecta orice combinație de intrări sau ieșiri care pot genera întreruperi. Dacă, de exemplu, cuvîntul mască de mai sus ar fi fost:

0	1	0	1	0	1	1	0
---	---	---	---	---	---	---	---

atunci o cerere de întrerupere ar fi apărut și dacă bitul A7 (test special) din registrul de ieșire ar fi fost înscris.

Este posibil ca adresele porturilor să fie de exemplu :  $E0_H = \text{date port A}$  ;  $E1_H = \text{date port B}$  ;  $E2_H = \text{control port A}$  ;  $E3_H = \text{control port B}$ .

Toate adresele sînt codificate mai sus în hexazecimal. Valorile sînt convenabile, deoarece linia de selecție  $B/\overline{A}$  este uzual conectată la linia de adrese A0 iar linia de selecție  $C/\overline{D}$  la linia de adrese A1. Linia de selecție a circuitului,  $\overline{CE}$ , se decodifică din liniile de adresă A2—A7 ale unității centrale. Dacă se utilizează un număr restrîns de periferice, nu este necesar un decodificator pentru linia  $\overline{CE}$ , fiind posibilă conectarea ei la una dintre liniile A2—A7.

În continuare se vor prezenta exemple de interfațare cu dispozitive de intrare/ieșire, incluzînd și subrutinele care efectuează operațiile dorite și care sînt incluse în programul monitor cu o lungime de 2 kiloocteți pentru sistemul Z80 și care va fi listat ca anexă la capitolul 9.

### 3. Conectarea unui cititor de bandă de hîrtie perforată, LB—50

Cititorul de bandă de hîrtie perforată LB—50 (produs de I.E.P. București) permite citirea a 50 de caractere pe secundă și derularea rapidă a benzii, în ambele sensuri.

Pentru utilizare se pot folosi și numai semnalele NSTSP, NVAL, liniile de date și alimentările, restul liniilor cititorului rămînînd neconectate.

Semnalul NVAL indică faptul că cititorul este pregătit pentru o comandă de citire și din ce moment este informația disponibilă la conector. Comanda de citire poate fi trimisă lectorului de bandă dacă NVAL este la 1 logic și data poate fi preluată după ce NVAL este 0 logic. Durata de 2—15 ms a semnalului NVAL este reglabilă, asigurînd viteza de citire de 50 de caractere pe secundă.

Semnalul NSTSP trebuie menținut în repaus la 1 logic și se pune la 0 pentru o comandă de citire, stare în care trebuie să rămîna pînă la apariția lui  $NVAL = 0$ . NSTSP acționează pe front negativ, deci menținerea sa la 0 logic nu permite decît o singură citire și deplasarea benzii perforate cu un singur caracter. Diagrama în timp care reprezintă semnalele NSTSP și NVAL este dată în figura 6.20. Conectarea cititorului de bandă LB—50 se poate face direct la liniile de date și conversație ale portului



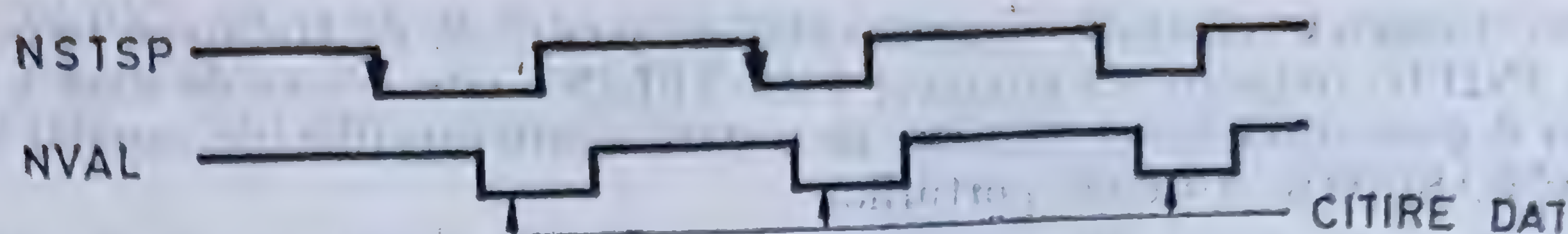


Fig. 6.20 Diagramă în timp pentru acționarea cititorului de bandă perforată LB-50.

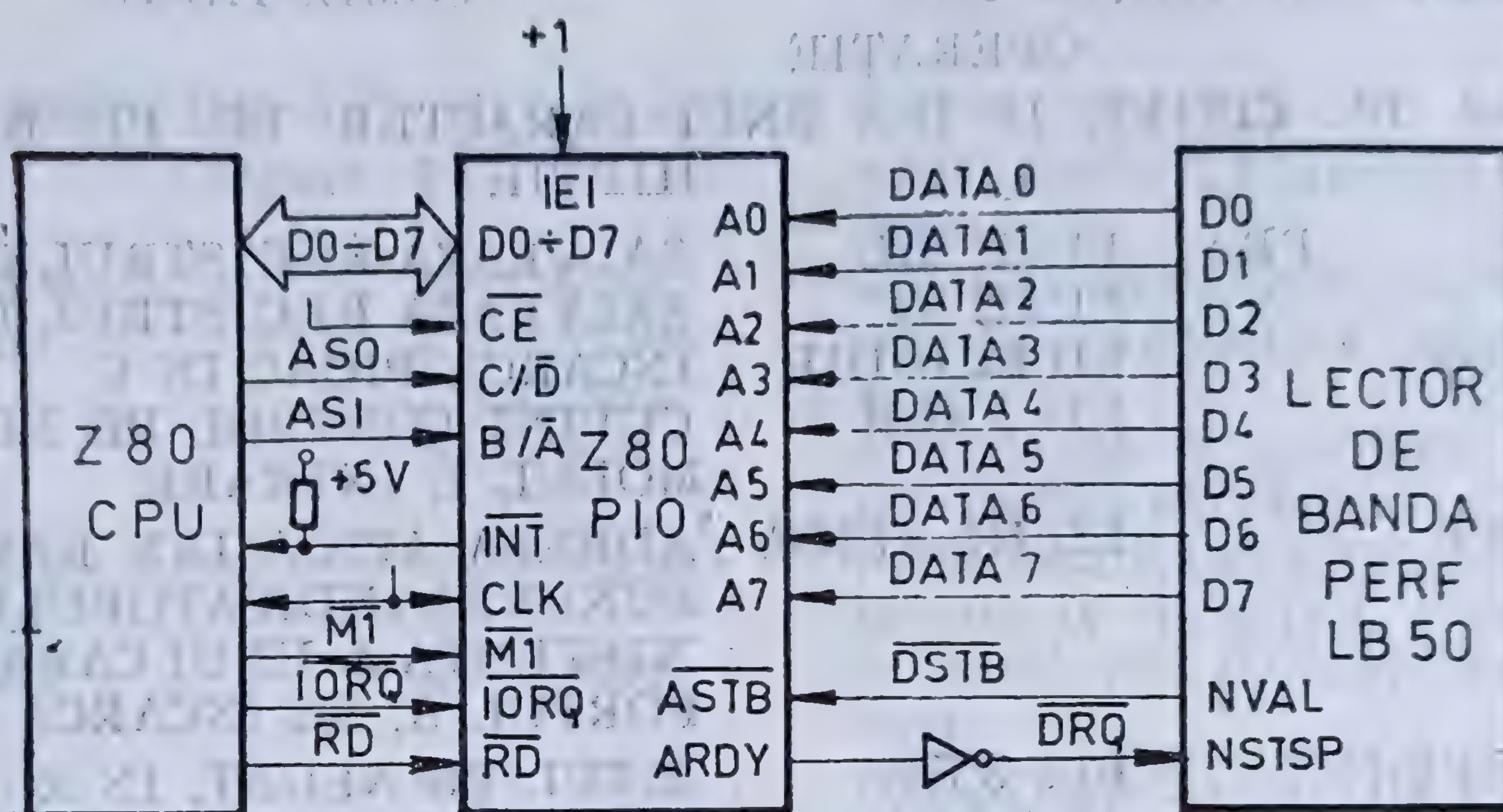


Fig. 6.21 Conectarea cititorului de bandă perforată LB 50 la circuitul Z80 PIO.

A (figura 6.21), cu excepția semnalului de avans al benzii, care trebuie inversat, fiind astfel posibilă citirea benzilor de hîrtie la comanda specializată a unui program monitor, descris în capitolul 9.

#### 4. Conectarea unui cititor de bandă de hîrtie perforată FS 330

Cititorul de bandă de hîrtie FS 330 (fabricație RSC) permite citirea a 300 de caractere/secundă și nu necesită tensiuni de alimentare externe. Logica de citire a perforațiilor de pe banda de hîrtie este însă inversă față de cea a cititorului LB-50. De aceea, pentru a citi o bandă pregătită pentru cititorul LB-50 cu același program de citire, liniile de date provenind de la cititorul FS 330 trebuie inversate prin hard (altfel ar trebui complementate prin soft, după citire). Pentru conectare se pot utiliza numai liniile de date semnalele SC1, AC și de masă. Conectarea la circuitul PIO portul A este prezentată în figura 6.22. Semnalul AC (start) este echivalentul lui NSTSP de la LB-50 iar SC1 (informația pregătită), este echivalentul lui NVAL. De notat că în conectorul cititorului, pinul b12, conexiunea comună a rezistențelor canalelor de date, trebuie menținut la o tensiune pozitivă (în funcție de nivelul dorit pe liniile de date) care poate fi cea de +5V de pe pinul a12.

În continuare, se prezintă subrutina de citire a unui caracter de pe banda de hîrtie perforată, din cadrul unui program monitor de 2 kiloocteți pentru sistemul Z80. Subrutina PR constituie un driver pentru cititorul de bandă perforată și utilizează circuitul PIO,

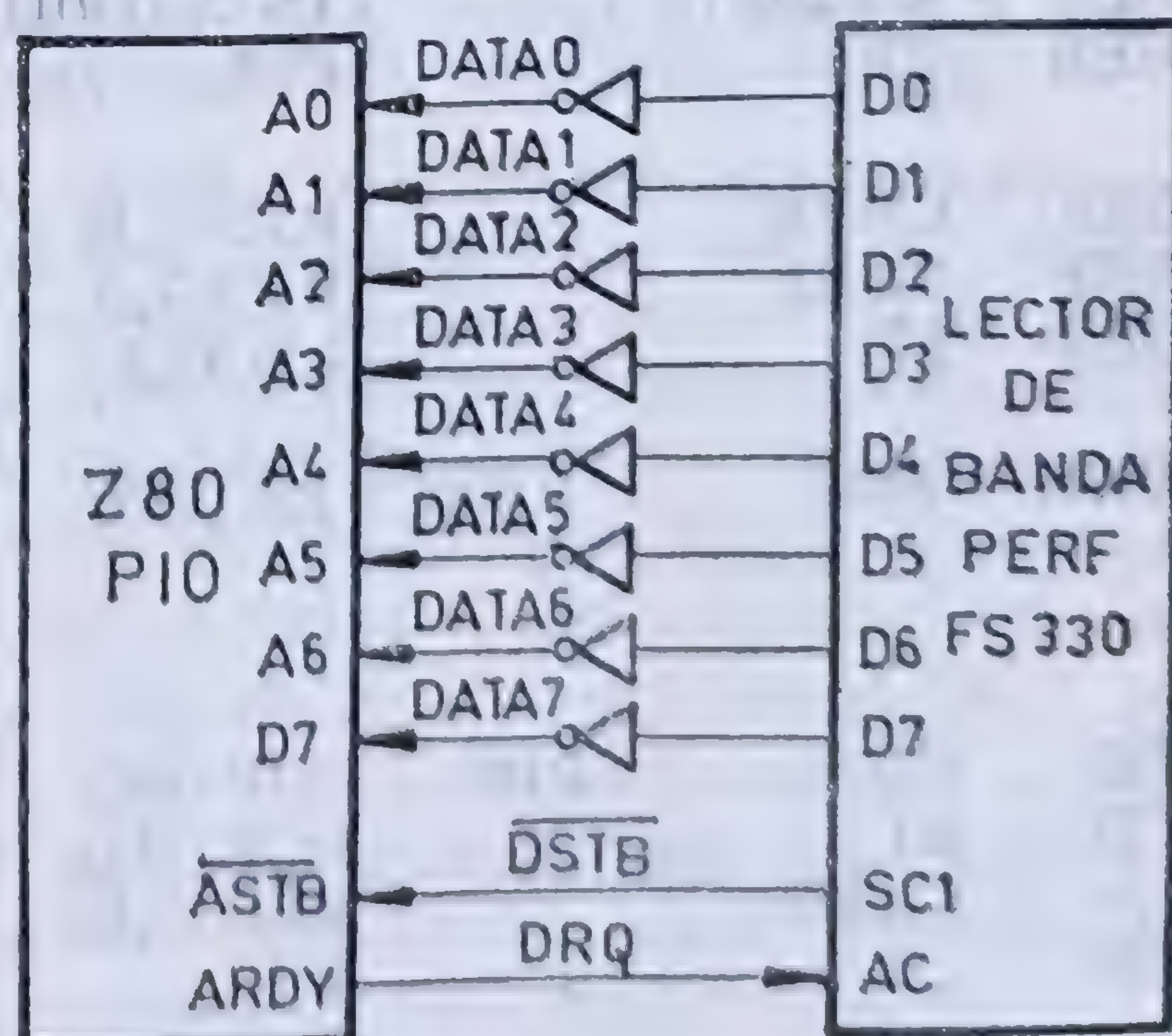


Fig. 6.22 Conectarea cititorului de bandă de hîrtie perforată FS330 la circuitul Z80 PIO.



canalul A. Unitatea centrală funcționează în modul 2 de tratare a întreruperilor. Subrutina INIPIO inițializează circuitul PIO. TBLINT este adresa de bază a tabelului în care sînt depuse adresele subrutinelor de tratare a întreruperilor pe canalul A, SPIOA și respectiv B, SPIOB. Adresele porturilor

PIOAD (port A, date), PIOAC (port A, control), PIOBD (port B, date), PIOBC (port B, control), sînt respectiv D0, D1, D2, D3<sub>H</sub>.

ADRESA CODUL ETICHETA COD COMENTARIU

# Operație

; SUBROUTINA DE CITIRE IN D A UNUI CARACTER DE PE BANDA DE HÎRTIE

E6C6	E5	PR:	PUSH HL	; SALVEAZA REGISTRUL DUBLU HL
E6C7	C5		PUSH BC	; SALVEAZA REGISTRUL DUBLU BC
E6C8	01D100		LD BC,00D1H	; INCARCA PIOAC IN C
E6CB	3E4F		LD A,4FH	; CUVINT CONTROL DE MOD IN A—
				; MODUL 1, INTRARE
E6CD	2121FF		LD HL,APIOA	; ADRESA APIOA DIN RAM PT. DE-
				; PUNEREA INDICATORULUI DE E-
				; XISTENTA A UNUI CARACTER IN
				; PORTUL A, SE INCARCA IN HL
E6D0	CB5B		BIT 3,E	; BITUL E3 NEGAT, IN Z
E6D2	2805		JR Z,PR1	; SALT LA PR1 DACA E3=0
E6D4	CD2EE7		CALL INIPIO	; INITIALIZARE PIO
E6D7	DBD0		IN A,(D0H)	; CITIRE FALSA DE DATE PT.A
				; FIXA ARDY LA 1
E6D9	FB	PR1:	EI	; VALIDEAZA INTRERUPERILE
E6DA	CB7E	PR4:	BIT 7,(HL)	; BITUL(HL)7 NEGAT IN Z
E6DC	2006		JR NZ,PR2	; SALT LA PR2 DACA(HL)7=1
E6DE	CB7B		BIT 7,E	; BITUL E7 NEGAT IN Z
E6E0	200C		JR NZ,PR3	; SALT LA PR3 DACA E7=1
E6E2	18F6		JR PR4	; SALT LA PR4 DACA E7=0, PENTRU
				; ASTEPTARE CARACTER
E6E4	3600	PR2:	LD (HL),00H	; DEPUNE 0 LA ADRESA APIOA
				; (EXISTA CARACTER)
E6E6	DBD0		IN A,(D0H)	; CITESTE IN A DATELE DIN PIO AD
E6E8	2F		CPL	; OCTETUL CITIT ESTE COMPLEMEN-
				; TAT
E6E9	CBBB		RES 7,E	; STERGE BITUL E7
E6EB	CBBF		RES 7,A	; STERGE BITUL A7, DE PARITATE
E6ED	57		LD D,A	; TRANSFERA CONTINUTUL LUI A
				; IN D
E6EE	1832	PR32:	JR PP6	; SALT LA ADRESA PP6
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
E722	C1	PP6:	POP BC	; REFACE REGISTRUL DUBLU BC
E723	E1		POP HL	; REFACE REGISTRUL DUBLU HL
E724	CB9B		RES 3,E	; STERGE BITUL E3
E726	C9		RET	; REVINE IN PROGRAMUL PRINCI-
.	.	.	.	; PAL
.	.	.	.	.
.	.	.	.	.



; TABEL CU ADRESELE SUBROUTINELOR DE TRATARE A ÎNTRERUPERILOR

E728 42E7 TBLINT: DW SPIOA ; ADRESA SUBROUTINEI DE TRATARE A ÎNTRERUPERILOR PT. PORTUL A

E72A 4CE7 DW SPIOB ; ADRESA SUBROUTINEI PT. PORTUL B

; SUBROUTINA DE INITIALIZARE PIO

E72E F3 INIPIO: DI ; ANULEAZA VALIDAREA ÎNTRERUPERILOR

E72F EDF9 OUT (C),A ; CUVINT CONTROL MOD 1 IN C

E731 3E28 LD A,LOW ; OCTET INFERIOR AL TBLINT IN (TBLINT)

E733 80 ADD A,B ; ADUNA B SI OBTINE VECTORUL DE ÎNTRERUPERI

E734 ED79 OUT (C),A ; VECTOR DE ÎNTRERUPERI INSCRIS IN PIOAC (PORTUL A CONTROL)

E736 3E83 LD A,83H ; CUVINT VALIDARE ÎNTRERUPERI PIO

E738 ED79 OUT (C),A ; INSCRIE CUVINTUL IN PIOAC

E73A 3600 LD (HL),00 ; DEPUNE 0 LA ADRESA APIOA SAU APIOB (ADRESA (HL))

E73C 3EE7 LD A,HIGH ; OCTET SUPERIOR TBLINT IN A (TBLINT)

E73E ED47 LD I,A ; INCARCA IN REGISTRUL I

E740 FB EI ; VALIDEAZA ÎNTRERUPERILE

E741 C9 RET ; REVINE IN PROGRAMUL PRINCIPAL

; SUBROUTINA DE TRATARE A ÎNTRERUPERILOR LA RECEPTIA UNUI CARACTER LA PORTUL A

E742 F5 SPIOA: PUSH AF ; SALVEAZA REGISTRUL DUBLU AF

E743 3EFF LD A,FFH ; INCARCA FFH IN A

E745 3221FF LD (APIOA),A ; INCARCA FFH IN APIOA, IN RAM

E748 F1 S1: POP AF ; REFACE REGISTRUL DUBLU AF

E749 FB EI ; VALIDEAZA ÎNTRERUPERILE

E74A ED4D RETI ; REVINE DIN ÎNTRERUPERE

Se observă că apelul acestei subrutine are ca efect următoarele acțiuni: se salvează HL și BC; se încarcă în B valoarea 00<sub>H</sub> și în C valoarea D1<sub>H</sub>, care este adresa portului de control al canalului A din PIO; se pregătește cuvântul de control de mod pentru modul 1 (intrare) în registrul A și adresa APIOA de depunere în RAM a unui indicator de existență a unui caracter la portul A din PIO. Dacă PIO a citit un octet de la periferic, subrutina de întrerupere SPIOA înscrie valoarea FF în locația cu adresa în APIOA=HL, deci (HL)<sub>7</sub> = 1, ceea ce va indica faptul că portul PIO A conține o dată validă. După încărcarea adresei APIOA în HL, se testează bitul E<sub>3</sub> al registrului E, care are valoarea 1 la intrarea în program (la prima parcurgere a lui), arătând că circuitul PIO trebuie inițializat. Dacă E<sub>3</sub>=1, se apelează subrutina INIPIO de inițializare a circuitului PIO, care dezactivează întreruperile, înscrie cuvântul de control pentru modul 1 al portului A, înscrie vectorul de întreruperi în portul PIO (octetul inferior TBLINT), un cuvânt de validare a întreruperilor de la PIO în portul A control, depune 00<sub>H</sub> la adresa APIOA (astfel (HL)<sub>7</sub>=0, arătând că încă nu există



o dată validă în portul A de date), încarcă octetul superior al adresei TBLINT în registrul I al microprocesorului, validează întreruperile și revine în programul principal. După execuția acestei subrutine, orice întrerupere care apare de la portul A, în cazul înscrierii în portul A a unei date de către periferic, are ca efect plasarea de către circuitul PIO a vectorului de întreruperi pe liniile de date, care, combinat cu conținutul registrului I, formează adresa TBLINT de unde microprocesorul va prelua adresa de 2 octeți a subrutinei de tratare a întreruperilor de la portul A, numită SPIOA.

După execuția subrutinei INIPPIO, la prima parcurgere a programului PR, dacă  $E_3=1$ , sau fără execuția ei, dacă  $E_3=0$ , se testează bitul  $(HL)_7$  pentru a vedea dacă portul A conține o dată validă. Dacă  $(HL)_7=0$ , data nu există și se testează bitul  $E_7$  al registrului E. Dacă la intrarea în program s-a inițializat  $E_7=1$  (în programul apelant), nu se mai face nici o încercare de a aștepta o dată validă la portul A, se efectuează un salt la adresa PR3, în continuare la PP6, unde se refac registrele BC, HL, se face  $E_3=0$  și se revine în programul care a apelat subrutina PR. Dacă  $E_7=0$ , se efectuează un salt la adresa PR4, așteptând într-o buclă ca  $(HL)_7$  să devină 1, prin înscrierea unei date în portul A.

Dacă la testarea bitului  $(HL)_7$  s-a găsit  $(HL)_7=1$ , atunci o dată validă se află în portul A. Se efectuează un salt la adresa PR2, se citește data (caracter ASCII), se complementează, se face  $E_7=0$ , se șterge bitul de paritate al datei ( $A_7$  din A), se încarcă data în registrul D, se efectuează un salt la adresa PF6, se refac BC, și HL, se face  $E_3=0$  și se revine în programul care a apelat subrutina PR. La revenire, data citită se află în registrul D, iar bitul  $E_3$  este șters,  $E_3=0$ .

## 5. Conectarea unui perforator de bandă de hîrtie P 50

Perforatorul de bandă de hîrtie P 50 (fabricație I.E.P. București), permite perforarea a 50 de caractere pe secundă. Pentru conectare la un sistem cu microprocesor Z80, se poate utiliza un circuit Z80 PIO, conectat direct la perforatorul de bandă cu excepția unui inversor necesar pe linia CP a perforatorului.

Semnalul CP, de comandă a perforării, este activ pe frontul negativ. Datele fixate pe portul PIO trebuie să fie stabile pînă la apariția semnalului de confirmare RBPER de la perforator, activ pe 0 și cu o durată de aproximativ 45  $\mu s$ .

Modul de conectare al perforatorului de bandă de hîrtie P 50 la un sistem Z80 prin intermediul unui circuit Z80 PIO este prezentat în figura 6.23. Pentru a asigura compatibilitatea benzii perforate în modul de conectare de mai sus cu cititoarele de bandă conectate ca în figurile 6.21 sau 6.22, utilizînd subrutinele de citire și perforare prezentate în acest capitol, semnalul de alegere a logicii de perforare, LOG, trebuie să

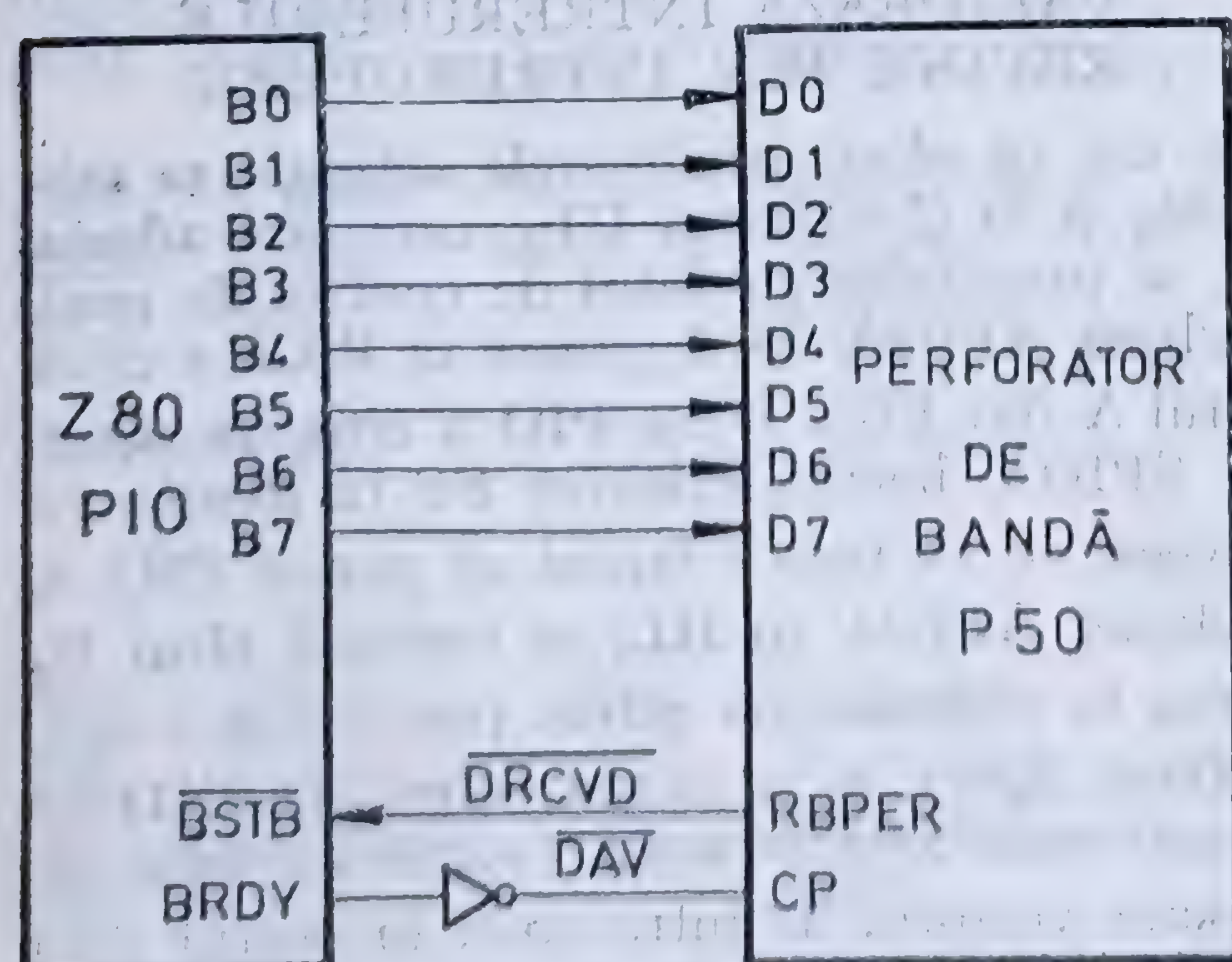


Fig. 6.23 Conectarea perforatorului de bandă de hîrtie P 50 la circuitul Z80 PIO.

fie conectat la 0 logic (se conectează pinul 21 la pinul 24 al conectorului, obținînd astfel perforație pe bandă pentru „1” logic în cuvîntul (de date).

Subrutina de perforare a unui caracter pe banda de hîrtie (driver pentru perforatorul de bandă), care utilizează circuitul PIO, canalul B, este listată în continuare. Unitatea centrală funcționează în modul 2 de întreruperi. Subrutina INIPPIO și tabelul de adrese TBLINT sînt cele din cazul subrutinei PR, prezentată anterior. Modul de acțiune al subrutinei PP este asemănător cu cel al subrutinei PR, fapt rezultat și din comentariile asociate fiecărei instrucțiuni.



ADRESA		CODUL ETICHETA		COD OPERATIE		COMENTARIU
; SUBROUTINA DE PERFORARE A UNUI CARACTER DIN D PE BANDA						
; DE HIRTIE						
E6FA	E5	PP:	PUSH HL	; SALVEAZA HL		
E6FB	2122FF		LD HL,APIOB	; ADRESA APIOB=FF22H DE INDI-		
				; CARE A CERERII DE TRANSMITE-		
				; RE A UNUI CARACTER PRIN POR-		
				; TUL B, (CIND CONTINUTUL EI ARE		
				; D7=1), SE INTRODUCHE IN HL		
E6FE	C5		PUSH BC	; SALVEAZA BC		
E6FF	01D302		LD BC,02D3	; ADRESA PIOBC=D3H IN C SI 02H		
				; IN B, PENTRU CA IN INIPIO SA		
				; SE OBTINA VECTORUL DE INTRE-		
				; RUPERI 28H+02H=2AH (LA E72AH		
				; SE AFLA ADRESA E74CH A SUBRU-		
				; TINEI SPIOB)		
E702	F5		PUSH AF	; SALVEAZA AF		
E703	CB5B		BIT 3, E	; INCARCA E3 NEGAT IN Z		
E705	2807		JR Z,PP1	; SALT LA PP1 DACA E3=1 (NU SE		
				; FACE INITIALIZAREA)		
E707	3E0F		LD A,0FH	; CUVINT CONTROL MOD 0 PREGA-		
				; TIT IN A PENTRU PORTUL B		
E709	CD2EE7		CALL INIPIO	; APELEAZA SUBROUTINA DE INI-		
				; TIALIZARE PIO; INITIALIZEAZA		
				; PORTUL B IN MODUL 0, IESIRE DE		
				; OCTET, CU VECTOR DE INTRERU-		
				; PERE, CU VALIDAREA INTRERU-		
				; PERILOR; DEPUNE, 00H LA A-		
				; DRESA APIOB		
E70C	180B		JR PP3	; SALT LA PP3		
E70E	FB	PP1:	EI	; VALIDEAZA INTRERUPERI LA		
				; CPU		
E70F	CB7E	PP4:	BIT 7,(HL)	; BITUL (HL)7 NEGAT, INTRODUS		
				; IN Z		
E711	2006		JR NZ,PP3	; SALT LA PP3 DACA (HL)7=1 (SE		
				; CERE TRANSMITEREA UNUI CA-		
				; RACTER)		
E713	CB7B		BIT 7,E	; BITUL E7 NEGAT SE INTRODUCHE		
				; IN Z		
E715	200A		JR NZ,PP5	; SALT LA PP5 DACA E7=1 (NU MAI		
				; ASTEAPTA CEREREA DE TRANS-		
				; MISIE)		
E717	18F6		JR PP4	; SALT LA PP4 DACA E7=0 (ASTEAP-		
				; TA CEREREA DE TRANSMISIE)		
E719	36C0	PP3:	LD (HL),00H	; DEPUNE 00H LA ADRESA APIOB		
E71B	7A		LD A,D	; CARACTERUL DE PERFORAT SE		
				; TRANSFERA DIN D IN A		
E71C	0B		DEC BC	; C CONTINE ACUM D2H=PIOBD		
				; (ADRESA PORTULUI B, DATE)		
E71D	FD79		OUT (C),A	; INSCRIE CARACTERUL IN PORTUL		
				; B, REZULTIND PERFORAREA LUI		
E71F	CB7B		RES 7,E	; FACE E7=0		
E721	F1	PP5:	POP AF	; REFACE AF		
E722	C1	PP6:	POP BC	; REFACE BC		
E723	E1		POP HL	; REFACE HL		



```

E724 CB9B RES 3,E ; FACE E3=0
E726 C9 RET ; REVINE IN PROGRAMUL CARE A
; APELAT SUBROUTINA PP
; SUBROUTINA DE TRATARE A INTRERUPERILOR LA CEREREA DE
; TRANSMITERE A UNUI CARACTER PRIN PORTUL B
E74C F5 SPIOB: PUSH AF ; SALVEAZA AF
E74D 3EFF LD A,FFH ; INCARCA FFH IN A
E74F 3222FF LD (APIOB),A ; DEPUNE FF LA ADRESA APIOB;
; D7=1 ARATA CA SE CERE TRANS-
; MITEREA UNUI CARACTER
E752 18F4 JR SI ; SALT LA S1
;
E748 F1 SI: POP AF ; REFACE AF
E749 FB EI ; VALIDEAZA INTRERUPERILE LA
; CPU
E74A ED4D RETI ; REVINE DIN SUBROUTINA DE IN-
; TRERUPERE

```

## 6. Conectarea unei imprimante DZM-180 la sistemul Z80, prin intermediul unui circuit Z80 PIO

Figura 6.24 ilustrează conectarea imprimantei DZM 180 (fabricație R.P.P.) la circuitul PIO, portul A. Magistrala de date este prevăzută cu porți TTL, cu colector în gol, 7406 și cu inversoare 7404, în scopul protecției împotriva paraziților.

Monostabilul 74121 are rolul de a asigura ștergerea întârziată, comandată de semnalul STB, a semnalului RDY. Numerotarea din figură indică conexiunile care trebuie efectuate cu imprimanta matriceală DZM-180.

Se prezintă în continuare o soluție posibilă pentru programul de comandă a imprimantei. Se disting 3 subrutine: prima, INPIOA, servește pentru inițializarea portului A din circuitul PIO; a doua, SINTA este subrutina de tratare a întreruperilor care apar la portul A din PIO și pregătește ieșirea unui caracter; a treia, WRCHAR, determină înscrierea caracterului în portul A din PIO, după ce apare apelul imprimantei de a i se transmite un caracter. Din programul principal, care stabilește conținutul registrului de întreruperi I, inițializează circuitul PIO, canalul A, cu ajutorul subrutinei INPIOA și apelează subrutina WRCHAR pentru scrierea unui caracter la imprimantă, se prezintă doar un fragment. Există posibilitatea, ca prin extinderea cu câteva instrucțiuni a subrutinei WRCHAR, microprocesorul să nu parcurgă indefinit bucla de așteptare care începe la adresa AST până când imprimanta cere transmiterea unui caracter și SINTA șterge locația (TRGOL) pentru a indica acest fapt, ci să testeze doar o singură dată dacă se cere transmiterea caracterului și să revină, în caz contrar, pentru o nouă testare, după execuția altor operații (similar cu modul de acțiune, dependent de bitul E7, din subrutinele PR și PP prezentate anterior). În acest mod, unitatea centrală nu va mai fi ocupată tot timpul cu urmărirea imprimantei. Cererea acesteia de a i se transmite un caracter va fi înregistrată cu ajutorul subrutinei de tratare a întreruperii la portul A din circuitul PIO.

În programul prezentat, portul A, date are adresa PLOAD = 80<sub>H</sub>, iar portul A, control, PIOAC = 81<sub>H</sub>. Locația cu adresa TRGOL, indică faptul că circuitul PIO nu trebuie să transmită date spre imprimantă, prin orice valoare nenulă înscrisă în locație. Dacă aceasta conține 00<sub>H</sub>, imprimanta este gata să accepte un caracter de la PIO. Vectorul de întrerupere este OITI, octetul inferior al adresei TBLINT la care







```

OUT (C),A      ; VALIDEAZA INTRERUPERILE LA PORTUL
                ; A
XOR A          ; STERGE A
LD (TRGOL),A   ; DEPUNE 00H IN LOCATIA CU ADRESA
                ; TRGOL
POP BC         ; REFACE BC
POP AF         ; REFACE AF
RET           ; REVINE

; SUBROUTINA DE TRATARE LA INTRERUPERILOR LA PORTUL A
; DIN PIO, LA CEREREA IMPRIMANTEI DE TRANSMITERE A UNUI
; CARACTER; INSCRIE 00H IN LOCATIA CU ADRESA TRGOL PENTRU
; A INDICA ACEASTA CERERE
SINTA: EI      ; VALIDEAZA INTRERUPERILE
PUSH AF       ; SALVEAZA AF
XOR A         ; STERGE A
LD (TRGOL),A  ; DEPUNE 00H IN LOCATIA CU ADRESA TRGOL
POP AF        ; REFACE AF
RETI          ; REVINE DIN INTRERUPERE
; SUBROUTINA DE SCRIERE A UNUI CARACTER IN PORTUL A; LA
; INTRAREA IN SUBROUTINA, CARACTERUL SE AFLA IN REGISTRUL
; A DIN CPU
WRCHAR: PUSH AF ; SALVEAZA AF
AST: LD A,(TRGOL) ; CONTINUTUL LOCATIEI CU ADRESA TRGOL
                ; SE TRANSFERA IN A
AND A         ; POZITIONEAZA BITUL Z PENTRU A TESTA
                ; DACA A ESTE 0
JR NZ,AST     ; SALT LA AST DACA (TRGOL) NU ESTE 0
POP AF        ; REFACE AF SI READUCE ASTFEL CARACTE-
                ; RUL DE TRANSMIS IN A
OUT (PIOAD),A ; SCRIE CARACTERUL DIN A IN PORTUL A PIO
LD (TRGOL),A  ; TRANSFERA CARACTERUL SI IN LOCATIA
                ; TRGOL PENTRU A INDICA PRIN CONTINUTUL
                ; NENUL AL EI CA IMPRIMANTA NU A CERUT
                ; INCA TRANSMITEREA URMATORULUI CARAC-
                ; TER
RET           ; REVINE
; PROGRAMUL PRINCIPAL
.
.
LD A,OSTI     ; OCTET SUPERIOR AL ADRESEI TBLINT, IN A
LD I,A        ; INCARCA OCTETUL IN REGISTRUL I
CALL INPIOA   ; INITIALIZEAZA PORTUL A DIN PIO
EI           ; VALIDEAZA INTRERUPERILE LA CPU
.
.
LD A,CAR      ; INTRODUC CARACTERUL DE TRANSMIS IN A
CALL WRCHAR   ; APELEAZA SUBROUTINA DE SCRIERE A UNUI
                ; CARACTER LA IMPRIMANTA
.
.
.

```



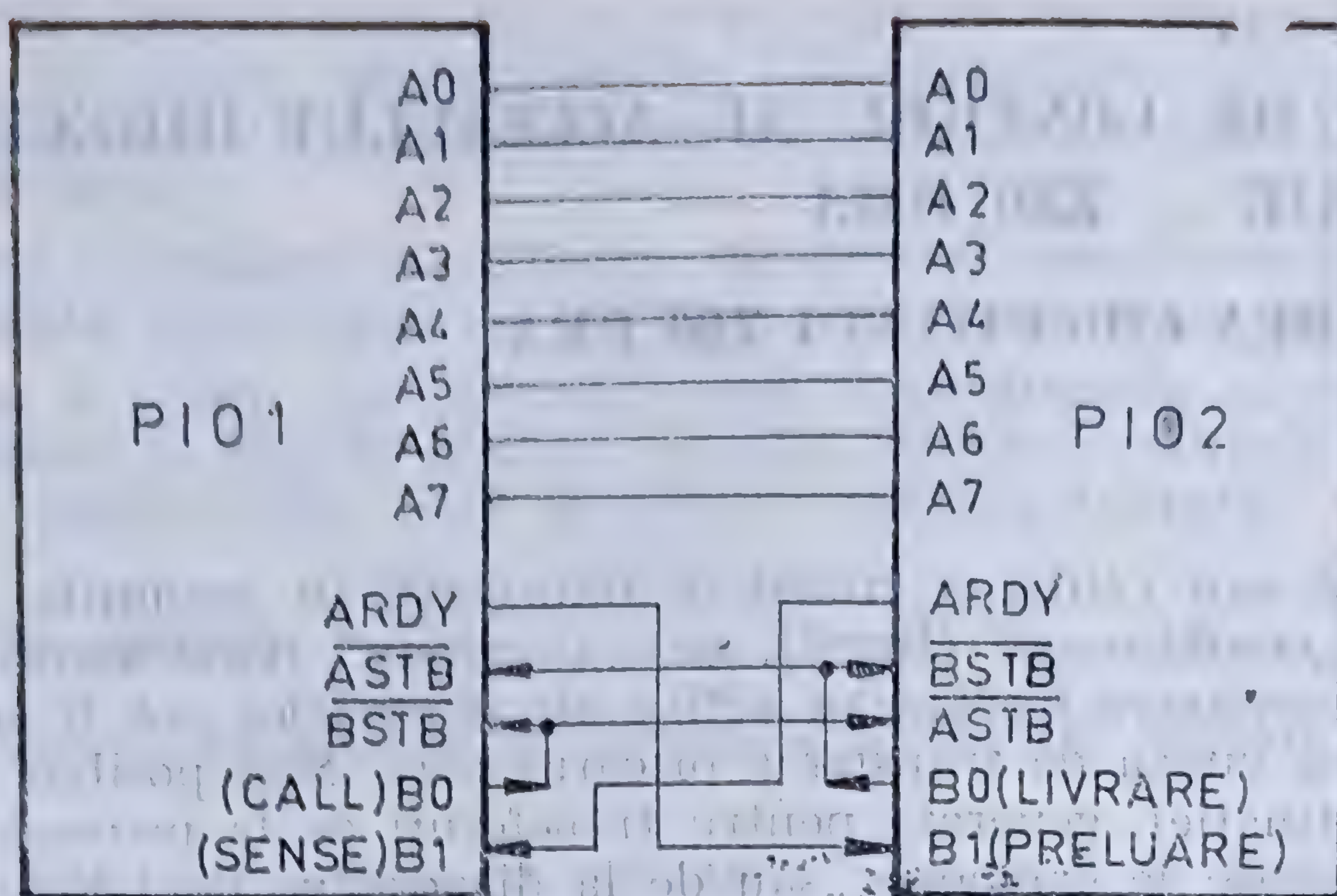


Fig. 6.25 Conectarea a două sisteme cu microprocesor prin intermediul circuitelor PIO.

; TABELA CU ADRESELE SUBROUTINELOR DE TRATARE A INTRE-  
; RUPERILOR

TBLINT: SINTAL ; OCTETUL INFERIOR AL ADRESEI SINTA  
SINTAH ; OCTETUL SUPERIOR AL ADRESEI SINTA

## 7. Interconectarea a două sisteme cu microprocesor

Transferul datelor de 8 biți între două sisteme cu microprocesor se poate realiza în paralel, prin intermediul a două circuite PIO, conectate ca în figura 6.25.

Semnalele READY și STROBE nu sînt conectate împreună, deoarece pe magistrala de date bidirecțională (A0—A7), nu pot fi plasate date decît de la un singur circuit PIO. Dacă registrele de intrare ar fi goale în ambele circuite, semnalele READY ar deschide ambele ieșiri. De aceea, semnalele ASTB nu sînt conectate la ieșirile BRDY; ele se conectează la unul dintre biții portului B, programat în modul de control (bitul notat LIVRARE). Acest semnal comandă și deschiderea propriului port. Semnalul ARDY al celuilalt port este conectat tot la un bit al portului B (PRELUARE).



## CAPITOLUL VII

# CIRCUITUL DE CONTROL AL ACCESULUI DIRECT LA MEMORIE — Z80 DMA

### 7.1. DESCRIEREA CIRCUITULUI Z80 DMA

Circuitul caută sau caută și transferă informații în modurile „cîte un octet” (Byte at a time), „condiționat” (Burst), sau „continuu” (Continuous mode). Lungimea ciclului și temporizarea fronturilor active ale semnalelor pot fi programate pentru a corespunde cu viteza de transfer a oricărui port. Sînt posibile adrese de port duale (sursă și destinație), generate pentru transferuri de la memorie la dispozitive de I/E, de la memorie la memorie, sau de la dispozitive de I/E la dispozitive de I/E. Adresele pot fi fixe sau incrementate/decrementate automat. Comanda pentru următoarea operație se poate încărca fără a deranja funcționarea curentă, prin registre pentru adresa de pornire, prevăzute cu tampon. Secvențe anterioare pot fi repetate în întregime în mod automat. Programarea funcțiilor se poate face extensiv, unitatea centrală poate citi complet starea caracterului, iar logica de cerere de magistrală și cea de priorități este implementată fără logică externă. Întreruperile se pot efectua cu modificarea vectorilor de întrerupere, iar interfațarea la magistrala sistemului se face direct, fără logică externă.

Circuitul Z80 DMA permite controlul și procesarea transferurilor de date, funcția de bază fiind de a realiza transferul de date independent de unitatea centrală, între două porturi, cu optimizarea vitezei de transfer, fără logică externă sau cu logică externă redusă, în sisteme cu magistrală de adrese de 16 biți și de date de 8 sau 16 biți.

Este posibilă căutarea de octeți sau de anumite combinații de biți într-un octet, fie simultan cu transferul, fie ca operație în sine.

Funcțiile circuitului Z80 DMA, care are trei posibilități de bază în funcționare, sînt reprezentate în figura 7.1. Acestea sînt: transfer de date între două porturi (memorie sau periferic de I/E); căutarea unui octet, mascabil, la un singur port în memorie sau la un periferic de I/E; transferuri combinate cu căutare simultană între două porturi.

În timpul unui transfer, circuitul DMA preia controlul magistralelor de date și de adrese. Datele sînt citite dintr-un port adresabil și înscrise în altul, octet cu octet. Porturile pot fi programate fie ca locații de memorie ale sistemului, fie ca porturi de I/E. Astfel, se pot transfera blocuri de date de la un periferic la altul, de la o zonă de memorie la alta, de la un periferic la memorie sau invers. În timpul unei operații de căutare (fără transfer), datele sînt citite dintr-un port sursă și sînt comparate octet cu octet cu conținutul unui registru intern al circuitului DMA, ce constituie un octet, programabil, de coincidență. Opțional, acest octet poate fi mascat, astfel

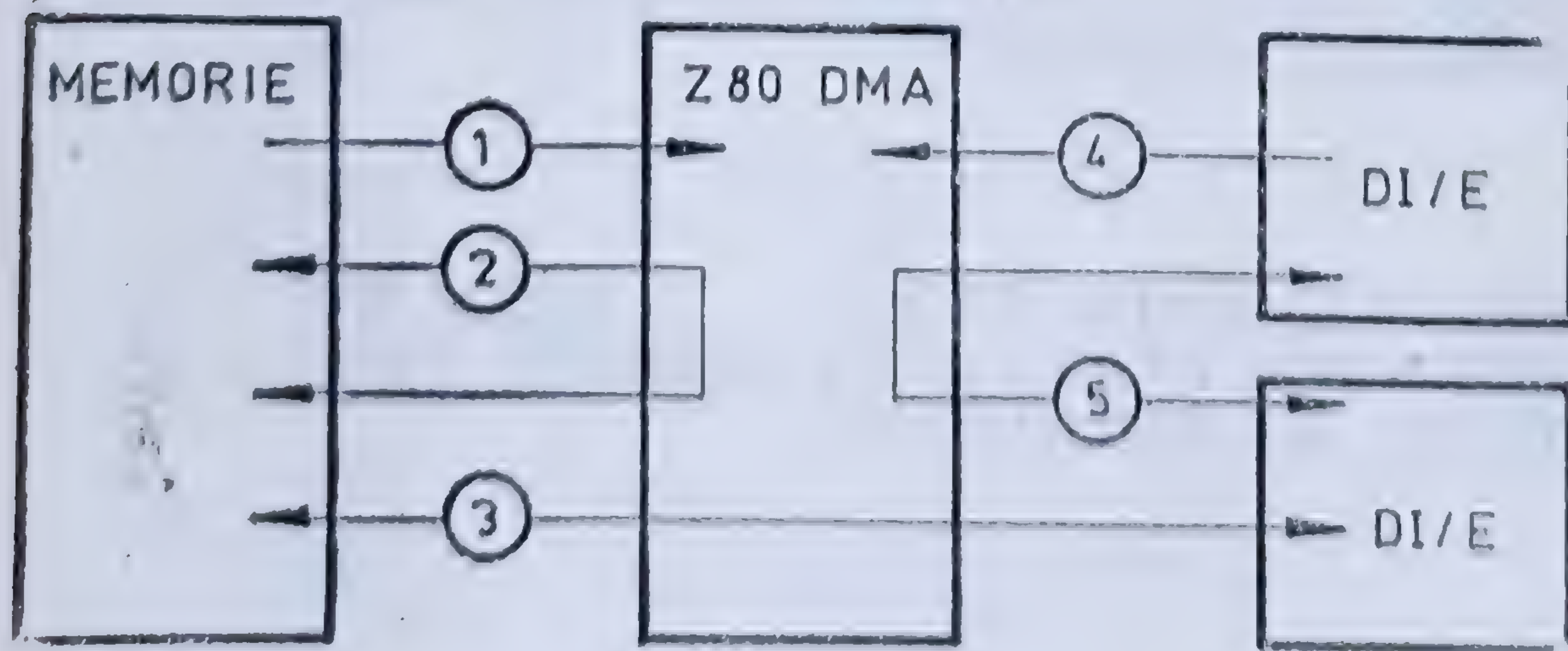


Fig. 7.1 Funcțiile circuitului Z80 DMA: 1 — Căutare în memorie; 2 — Transfer memorie-memorie (căutare opțională); 3 — Transfer memorie DI/E (căutare opțională); 4 — Căutare în DI/E; 5 — Transfer DI/E-DI/E (căutare opțională).



încît numai anumiți biți ai octetului de coincidență să fie comparați cu datele citite din port. Viteza de căutare ajunge pînă la 1,25 Mocteți/sec. cu circuitul Z80 DMA (semnalul de tact de 2,5 MHz), sau de 2 Mocteți/sec. cu circuitul Z80 A DMA (semnal de tact de 4 MHz).

În transferuri combinate cu căutare, datele sînt transferate între două porturi în timp ce se caută coincidența cu un octet care poate avea unii biți mascați.

Transferurile de date sau căutările pot fi programate să se oprească sau să determine întreruperi în diferite condiții. În plus, unitatea centrală poate citi o serie de biți de stare, programabili, care să reflecte condiția apărută.

### Moduri de funcționare

Circuitul poate fi programat să funcționeze în trei moduri de transfer și/sau căutare:

1. Cîte un octet: operațiile se execută pe cîte un octet de date; între două operații, magistralele sistemului sînt cedate unității centrale, fiind solicitate din nou pentru următoarea operație.

2. Condiționat: operațiile asupra datelor continuă pînă cînd o linie Ready a unui port, conectată la circuitul DMA, devine inactivă; circuitul DMA se oprește și cedează magistralele sistemului, după încheierea operației curente pe octet.

3. Continuu: operațiile asupra datelor continuă pînă la atingerea sfîrșitului blocului de date, cînd sînt cedate și magistralele sistemului; dacă o linie Ready a unui port devine inactivă înainte ca această situație să apară, circuitul DMA face o pauză pînă cînd linia Ready devine din nou activă.

În toate modurile, dacă un octet a fost citit în circuitul DMA, operațiile asupra lui continuă, în ordine, indiferent de starea altor semnale (inclusiv linia Ready a portului).

Datorită metodei foarte rapide, cu registre tampon, de a citi datele în circuitul DMA, operațiile asupra unui octet nu se termină pînă cînd este citit următorul octet. Aceasta înseamnă că lungimea totală a unui transfer sau a unei căutări într-un bloc trebuie să fie de 2 sau mai mulți octeți și că lungimea de bloc programată în circuitul DMA trebuie să fie cu 1 mai mică decît lungimea dorită (numărul va fi  $N-1$  dacă  $N$  este lungimea blocului).

### Cuvinte de comandă și de stare

Circuitul Z80 DMA are mai multe registre de control care pot fi înscrise și registre de stare care pot fi citite de către unitatea centrală.

Cuvintele de control pot fi înscrise în circuitul DMA oricînd acesta nu controlează magistralele sistemului, dar înscrierea unui octet de control în DMA îl dezactivează pînă cînd este din nou activat de o comandă specifică. Octeții de stare pot fi citiți de asemenea în orice astfel de moment, dar înscriind comanda de Citire a octetului de stare (Read Status Byte) sau comanda de Începere a secvenței de citire (Initiate Read Sequence), se dezactivează circuitul DMA.

Cuvintele de control ale circuitului DMA includ cele care au efect imediat de validare (activare), dezactivare, inițializare (reset), încărcare registrelor tampon pentru adresa de pornire, continuare, ștergere de numărătoare, ștergere de biți de stare etc. În plus, pot fi înscriși octeți de control care fixează modul de funcționare, incluzînd modul și clasa operației, configurația porturilor, lungimea de bloc, modul de schimbare a adreselor, octetul de coincidență și octetul de mascare al acestuia, vectorul de întreruperi, condiția de afectare a vectorului de către stare, numărarea de impulsuri, condiția de auto-restart, modul de acțiune al liniilor Wait și Ready și citirea de mască.



Registrele de stare care pot fi citite includ un octet de stare general reflectând linia Ready, sfârșitul de bloc, coincidența de octeți și condițiile de întrerupere și registre de 2 octeți pentru numărarea curentă de octeți, pentru adresa portului A și pentru cea a portului B.

### Ciclul variabil

Circuitul Z80 DMA are caracteristica unică de lungime programabilă a ciclului de funcționare, ceea ce permite adaptarea mai ușoară la cerințele particulare ale altor componente de sistem (rapide sau lente) și maximizează viteza de transfer a datelor. Logica externă pentru semnale de condiționare nu este necesară. Două aspecte sînt importante legat de ciclul variabil: în primul rînd, ciclurile (perioadele) de citire și de scriere asociate porturilor sursă și respectiv destinație, pot fi programate independent, la lungimi de 2, 3 sau 4 cicluri de tact,  $T$ , sau mai multe, dacă sînt necesare cicluri de așteptare, mărind sau micșorînd astfel viteza cu care se schimbă toate semnalele (figura 7.2); în al doilea rînd, cele 4 semnale ale fiecărui port, asociate cu transferul de date (cerere de I/E, cerere de memorie, citește și scrie) pot avea fiecare frontul activ (crescător) al perioadei active, terminat cu o jumătate de perioadă  $T$  mai devreme; aceasta dă flexibilitate și viteză, permițînd semnale Read sau Write mai scurte decît în modul obișnuit, devenind inactive înainte ca datele să înceapă să se schimbe.

### Generarea adreselor

Circuitul DMA generează două adrese de 16 biți pentru fiecare operație de transfer: o adresă pentru portul sursă și o adresă pentru portul de destinație. Fiecare adresă poate fi variabilă sau fixă. Adresele variabile pot fi incrementate sau decrementate, începînd de la adresa de start programată. Posibilitatea de a genera o adresă fixă elimină necesitatea existenței semnalelor de validare a porturilor de I/E.

Adresele de port sînt multiplexate pe magistrala de adrese a sistemului, după cum circuitul DMA citește date din portul sursă sau le scrie în portul de destinație. Adresa curentă a fiecărui port este păstrată în două registre de adresă de cîte 16 biți, care pot fi citite de către unitatea centrală.

### Auto-restart

Adresa de început a fiecărui port poate fi reîncărcată automat la sfârșitul unui bloc, opțiune care este selectată de bitul de control pentru Auto Restart. Numărătorul de octeți este șters cînd se reîncarcă adresa.

Posibilitatea de auto-restart scutește unitatea centrală de supraveghere prin soită a operațiilor repetitive, ca refreșcarca tuburilor cu raze catodice (CRT), sau altele. În plus, cînd unitatea centrală are acces la magistrale în timpul transferurilor de tip „cîte un octet” sau „condiționat”, în registrele tampon pot fi înscrise diferite

adrese de start în timpul transferurilor, determinînd Auto Restart-ul să înceapă la o nouă locație.

### Întreruperi

Circuitul Z80 DMA poate fi programat să întrerupă unitatea centrală în trei situații diferite:

1. Întrerupere pe Ready (înainte cererii de magistrală).
2. Întrerupere la apariția unei Coincidențe
3. Întrerupere la Sfârșit de Bloc.

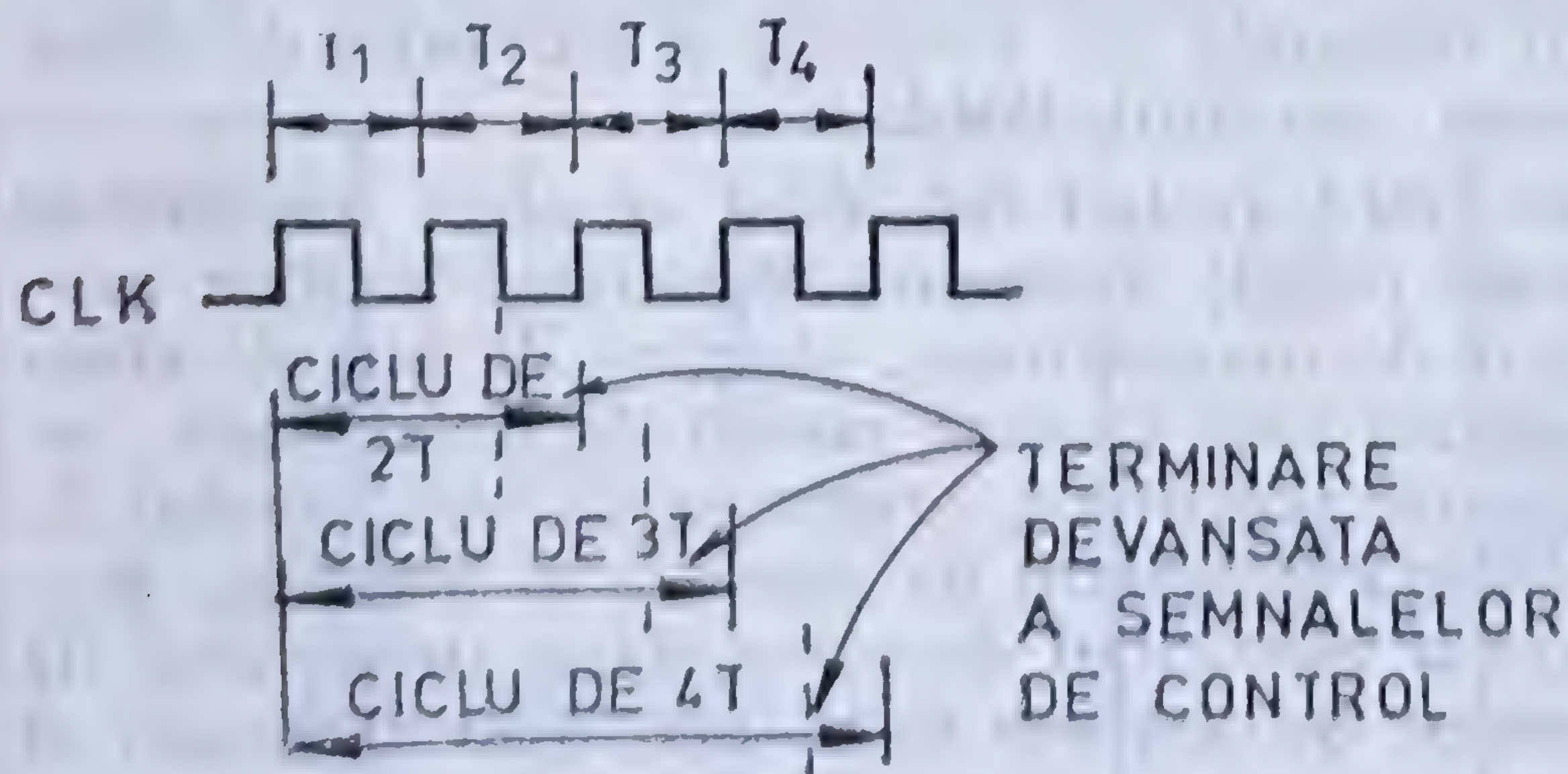


Fig. 7.2 Lungime de ciclu variabilă.



Oricare dintre aceste trei întreruperi determină înscriserea unui bit de stare, care așteaptă întreruperile și fiecare poate modifica, opțional, vectorul de întreruperi al circuitului Z80 DMA. Datorită constrîngerii date de registrul tampon, menționată mai sus, întreruperile pe Coincidență la Sfîrșit de Bloc sînt determinate de coincidența cu octetul imediat anterior ultimului octet din bloc.

Circuitul DMA se înscrisce în schema de întreruperi a familiei Z80, care asigură servirea rapidă a întreruperilor, pentru aplicații în timp real. Într-un sistem Z80 (figura 7.3) circuitul Z80 DMA își transferă vectorul de întreruperi de 8 biți, modificabil intern, spre unitatea centrală, care mai adaugă 8 biți pentru a forma adresa de memorie a tabelului care conține adresele rutinelor de întrerupere. În acest proces, controlul CPU este trecut direct la rutina de întrerupere, astfel încît următoarea instrucțiune executată după recunoașterea unei întreruperi este prima instrucțiune a rutinei de tratare a întreruperii.

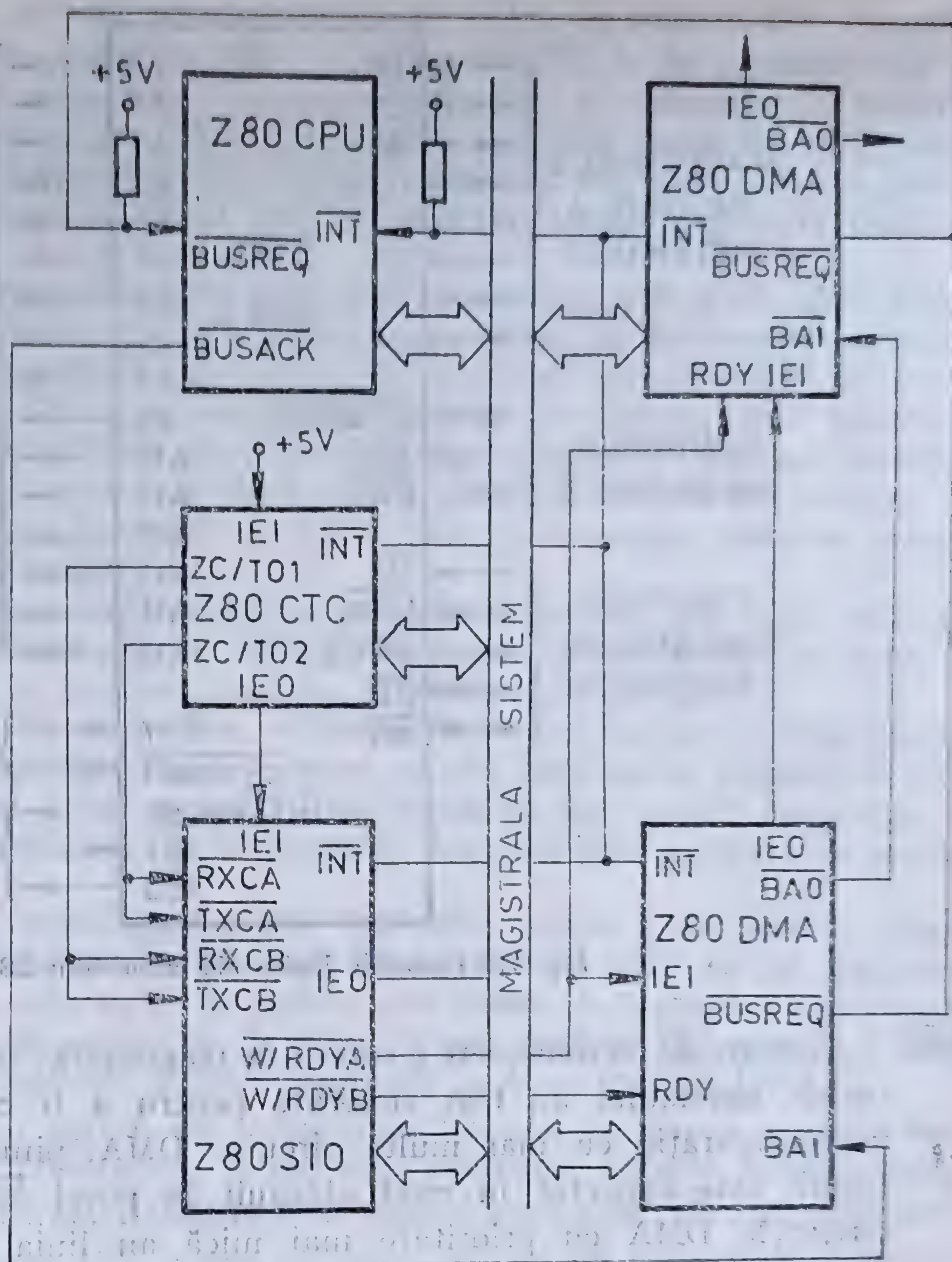


Fig. 7.3 Sistem Z80 tipic.

### Generarea impulsurilor

Dispozitivele externe pot ține evidența octeților transferați, utilizînd ieșirea de impulsuri a circuitului DMA, care furnizează semnale la intervale de 256 de octeți. Începutul intervalului poate fi decalat la pornire cu 1 pînă la 255 de octeți.

Linia de întrerupere asigură impulsurile într-un mod care nu permite interpretarea lor de către unitatea centrală ca și cereri de întrerupere, deoarece apar cînd liniile Bus Request (Cerere de magistrală) și Bus Acknowledge (Acceptarea cererii de magistrală) sînt ambele active.

### Descrierea conexiunilor circuitului Z80 DMA

Funcțiile logice ale circuitului Z80 DMA sînt reprezentate în figura 7.4. Descrierea rolului lor este dată în continuare.

A0—A15 — magistrala de adrese a sistemului, ieșiri cu trei stări; este utilizată pentru adresele generate de circuitul DMA atît spre portul sursă cît și spre portul destinație (memoria sistemului sau periferice de I/E).



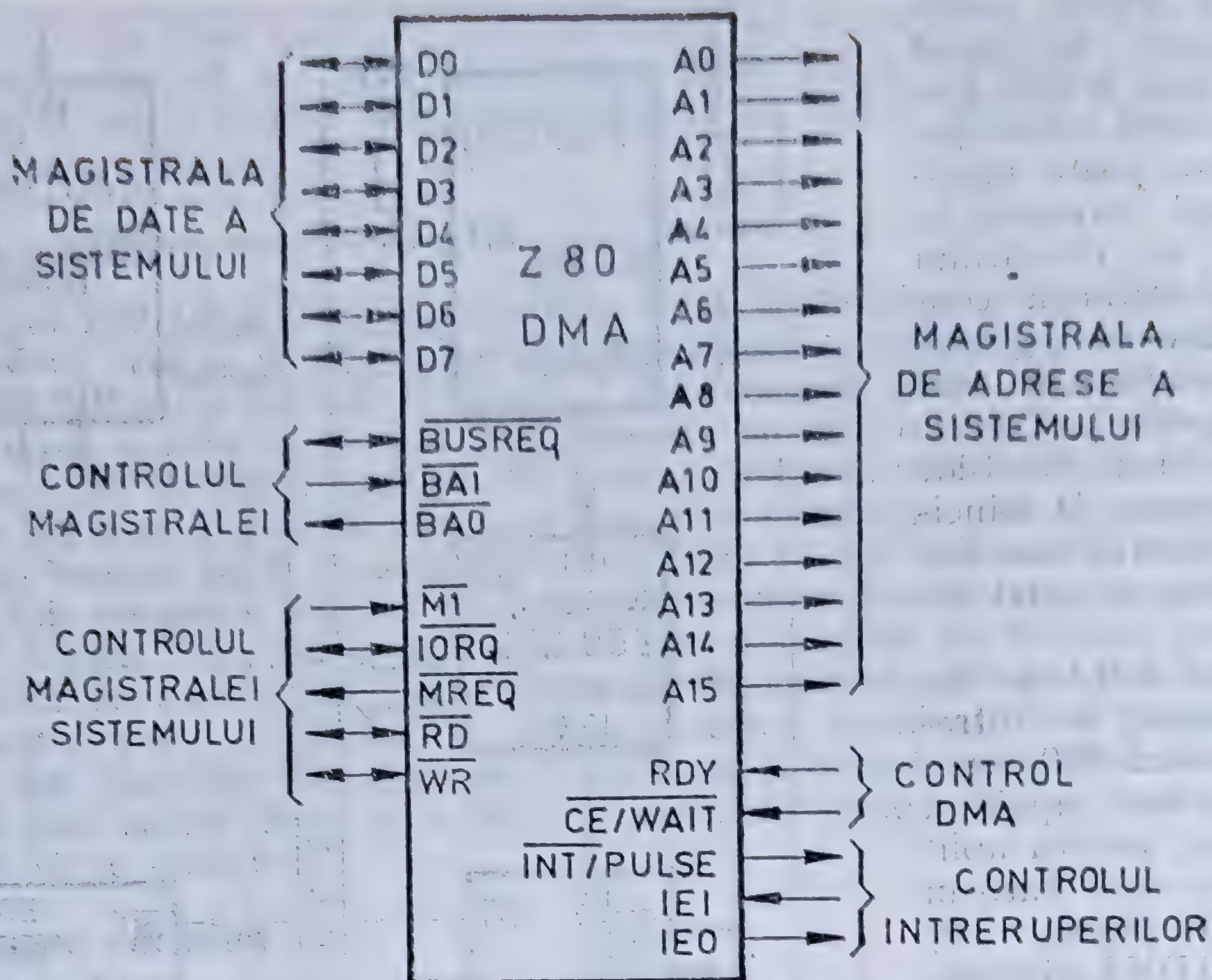


Fig. 7.4 Funcțiile logice ale circuitului Z80 DMA.

$\overline{\text{BAI}}$  — intrare de recunoaștere a cererii de magistrală; semnalizează faptul că magistralele sistemului au fost eliberate pentru a fi controlate de circuitul DMA; în configurație cu mai multe circuite DMA, pinul  $\overline{\text{BAI}}$  al celui mai prioritar circuit este conectat în mod obișnuit la pinul  $\overline{\text{BUSACK}}$  al unității centrale; circuitele DMA cu prioritate mai mică au linia  $\overline{\text{BAI}}$  conectată la linia  $\overline{\text{BAO}}$  a circuitului DMA cu prioritate mai mare.

$\overline{\text{BAO}}$  — ieșire de recunoaștere a cererii de magistrală; semnalizează, prin valoarea 0, în configurație cu mai multe circuite DMA, că nici un alt circuit DMA cu prioritate mai mare nu a cerut magistralele sistemului; liniile  $\overline{\text{BAI}}$  și  $\overline{\text{BAO}}$  formează un lanț de priorități pentru a decide asupra controlului magistralelor în sisteme cu mai multe circuite DMA.

$\overline{\text{BUSREQ}}$  — cerere de magistrală, bidirecțională, cu drenă în gol; ca ieșire, trimite spre unitatea centrală o cerere pentru controlul magistralei de adrese, a celei de date și a celei de control; ca intrare, în cazul conectării mai multor circuite DMA într-un lanț de priorități prin liniile  $\overline{\text{BAI}}$  și  $\overline{\text{BAO}}$ , sesizează momentul în care un alt circuit DMA cere magistralele și face ca acest circuit DMA să nu ceară magistralele pînă cînd circuitul menționat anterior nu termină operația pentru care a preluat controlul magistralelor; deoarece este o linie bidirecțională, nu pot exista tamponare între acest circuit DMA și oricare altul; poate exista totuși un tampon între acest circuit și unitatea centrală, pentru că linia este unidirecțională între circuitul DMA și unitatea centrală; un rezistor se conectează între acest pin și tensiunea de alimentare de +5 V.

$\overline{\text{CE/WAIT}}$  — intrare pentru selecția circuitului/așteptare; funcționează în mod obișnuit ca linie de selecție a circuitului, dar poate fi programată să servească și ca funcție  $\overline{\text{WAIT}}$ ; ca linie  $\overline{\text{CE}}$  de la unitatea centrală, devine activă cînd  $\overline{\text{WR}}$  și  $\overline{\text{IORQ}}$  sînt active și adresa de 1/16 de pe magistrala de adrese a sistemului este



adresa circuitului DMA, permițând un transfer de octet de control sau comandă de la unitatea centrală la circuitul DMA. Ca linie  $\overline{\text{WAIT}}$  de la memorie sau de la dispozitivele de I/E, determină inserarea de stări de așteptare în ciclurile de funcționare ale circuitului DMA, încetinind acțiunile acestuia la o viteză convenabilă memoriei sau dispozitivelor de I/E, după ce circuitul DMA a primit un semnal de cedare a magistralelor (bus request acknowledge) de la unitatea centrală.

**CLK** — intrarea de tact a sistemului; este semnalul standard de 2,5 MHz (Z80 DMA) sau de 4 MHz (Z80 A DMA); pentru valori mai reduse ale frecvenței semnalului de tact, cerințele dinamice și de nivele de tensiune specifice sînt de obicei satisfăcute de o poartă TTL, cu colectorul în gol; în sisteme cu o frecvență de tact mai mare trebuie utilizat un driver de tact pentru a satisface cerințele de  $V_{IH}$  și de timp de trecere de la 0 la 1; în toate cazurile un rezistor de 10 k $\Omega$  trebuie să fie prevăzut la sursa de +5 V pentru a asigura puterea corectă cînd circuitul DMA este inițializat.

**D0—D7** — magistrala de date bidirecțională a sistemului, cu trei stări, utilizată pentru transferul comenzilor de la unitatea centrală, a cuvintelor de stare de la DMA și a datelor de la memorie sau de la perifericele de I/E.

**IEI** — intrare de validare a întreruperilor, utilizată împreună cu ieșirea IEO pentru a forma un lanț de priorități cînd în sistem există mai multe dispozitive care pot cere întreruperi; un 1 pe această linie arată că nici un alt dispozitiv cu prioritate mai mare nu este servit de unitatea centrală într-o rutină de servire a unei întreruperi.

**IEO** — ieșire de validare a întreruperii; este utilizată pentru a nu permite dispozitivelor cu prioritate mai mică să ceară întreruperi în timp ce un dispozitiv cu o prioritate mai mare este servit în cadrul unei rutine de tratare a întreruperii de către unitatea centrală; este de 1 logic numai dacă și IEI este la 1 logic și unitatea centrală nu servește o întrerupere de la acest circuit DMA.

**INT/PULSE** — ieșire pentru cerere de întrerupere/impuls, cu drenă în gol; linia servește pentru a cere o întrerupere la unitatea centrală; această anunță acceptarea întreruperii trecînd linia  $\overline{\text{IORQ}}$  la 0 în timpul unui ciclu  $\overline{\text{M1}}$ ; în mod obișnuit, este conectată la linia  $\overline{\text{INT}}$  a unității centrale, cu un rezistor la tensiunea pozitivă de alimentare și este de asemenea conectată la toate celelalte linii  $\overline{\text{INT}}$  din sistem; această linie poate fi utilizată și pentru generarea de impulsuri periodice spre un dispozitiv extern, dar numai cînd circuitul DMA are controlul magistralelor, deci cînd liniile unității centrale  $\overline{\text{BUSREQ}}$  și  $\overline{\text{BUSACK}}$  sînt ambele la 0 și unitatea centrală nu poate sesiza cererile de întrerupere.

**$\overline{\text{IORQ}}$**  — linie bidirecțională, cu trei stări, pentru cererea de intrare/ieșire; ca intrare, arată că jumătatea inferioară a magistralei de adrese conține o adresă validă de port de I/E pentru transfer de cuvinte de control sau de stare de la, sau spre unitatea centrală; adresa vizează acest circuit DMA, dacă linia  $\overline{\text{CE}}$  a lui și linia  $\overline{\text{RD}}$  sau  $\overline{\text{WR}}$  sînt active; ca ieșire, după ce circuitul DMA a preluat controlul magistralelor sistemului, arată că magistrala de adrese de 8 biți sau de 16 biți conține o adresă validă de port pentru un alt dispozitiv de I/E implicat în transferul de date realizat de circuitul DMA; cînd semnalele  $\overline{\text{IORQ}}$  și  $\overline{\text{M1}}$  sînt active simultan, se anunță recunoașterea unei întreruperi de către unitatea centrală.

**$\overline{\text{M1}}$**  — intrare care indică primul ciclu de mașină; atunci cînd este activă, se desfășoară un ciclu de mașină de aducere a codului operației instrucțiunii; semnalul este utilizat de circuitul DMA pentru a decodifica instrucțiunea de revenire din întrerupere (RETI, cu codul  $\text{ED4D}_{16}$ ), emisă de unitatea centrală; în timpul aducerii codului instrucțiunilor cu 2 octeți pentru codul operației,  $\overline{\text{M1}}$  este activ



când este adus fiecare octet; când  $\overline{M1}$  și  $\overline{IORQ}$  sînt active simultan, se anunță recunoașterea unei întreruperi.

$\overline{MREQ}$  — ieșire cu trei stări, pentru cerere de memorie; arată că magistrala de adrese conține o adresă validă pentru o operație de citire sau de scriere a memoriei; după ce circuitul DMA a preluat controlul magistralelor sistemului, linia  $\overline{MREQ}$  indică un transfer de date de la sau spre memorie, prin circuitul DMA.

$\overline{RD}$  — linie bidirecțională, cu trei stări, pentru citire; ca intrare arată că unitatea centrală vrea să citească un cuvînt de stare din registrele de citire ale circuitului DMA, dacă este activă simultan cu  $\overline{CE}$  și  $\overline{IORQ}$ ; ca ieșire, după ce circuitul DMA a preluat controlul magistralelor sistemului, arată o citire de memorie sau de port de I/E, controlată de circuitul DMA.

$\overline{RDY}$  — intrare, activă pe 0 sau 1, în funcție de programare, cu semnificația „gata”; această linie este utilizată de circuitul DMA cînd este gata pentru o operație de citire sau de scriere, un dispozitiv periferic asociat cu un port DMA; în funcție de modul de lucru al circuitului DMA (octet, condiționat sau continuu), linia  $\overline{RDY}$  controlează indirect activitatea circuitului DMA, făcînd linia  $\overline{BUSREQ}$  să treacă la 0 sau la 1.

$\overline{WR}$  — linie bidirecțională, cu trei stări, pentru scriere; ca intrare arată că unitatea centrală vrea să înscrie cuvinte de control sau de comandă în registrele de scriere ale circuitului DMA, dacă este activă simultan cu  $\overline{CE}$  și  $\overline{IORQ}$ ; ca ieșire, după ce circuitul DMA a preluat controlul magistralelor sistemului, arată că are loc o scriere a unei locații de memorie sau a unui port de I/E, sub controlul circuitului DMA.

### Structura internă

Structura internă a circuitului Z80 DMA include circuite de recepție și de transmisie (driver) pentru interfațare cu o magistrală de date de 8 biți, cu o magistrală de adrese de 16 biți și cu linii de control de sistem (figura 7.5).

Într-un sistem Z80, circuitul DMA poate fi conectat direct la pinii analogi ai unității centrale, fără registre tampon adiționale, cu excepția liniei  $\overline{CE}/\overline{WAIT}$  (figura 7.6).

Magistrala internă a circuitului Z80 DMA realizează interfațarea cu magistrala de date a sistemului și deservește toată logica și registrele circuitului. Adresele generate de această logică pentru porturile A și B (sursă și destinație) ale canalului de transfer DMA, sînt multiplexate pe magistrala de adrese a sistemului;

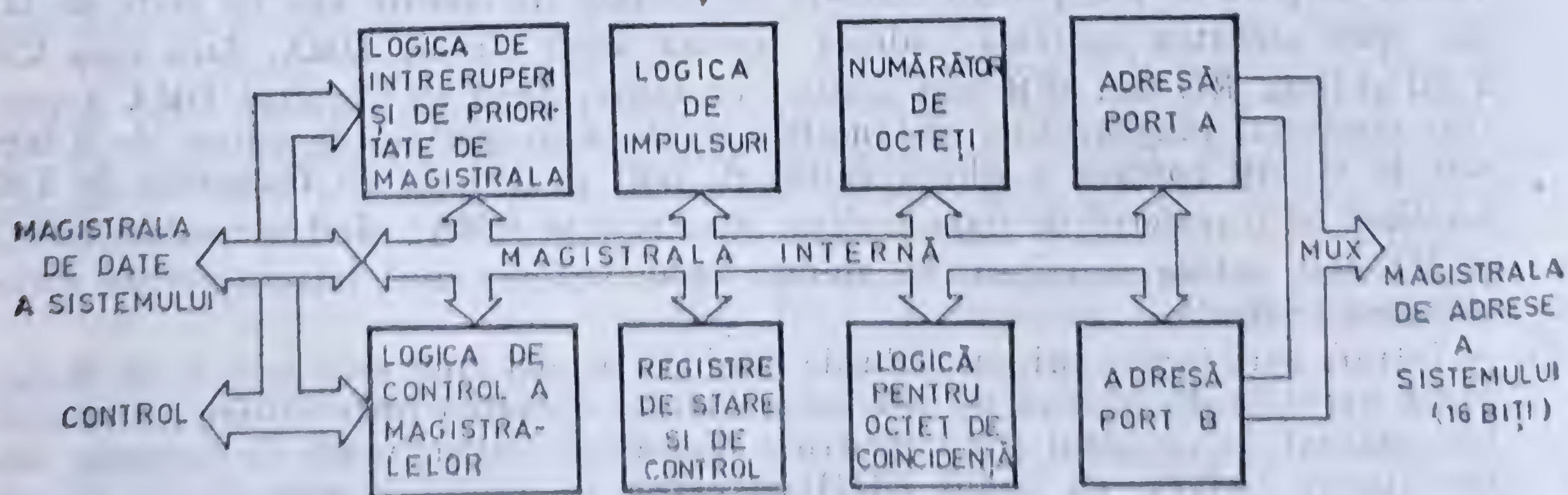


Fig. 7.5 Diagrama bloc a circuitului Z80 DMA.



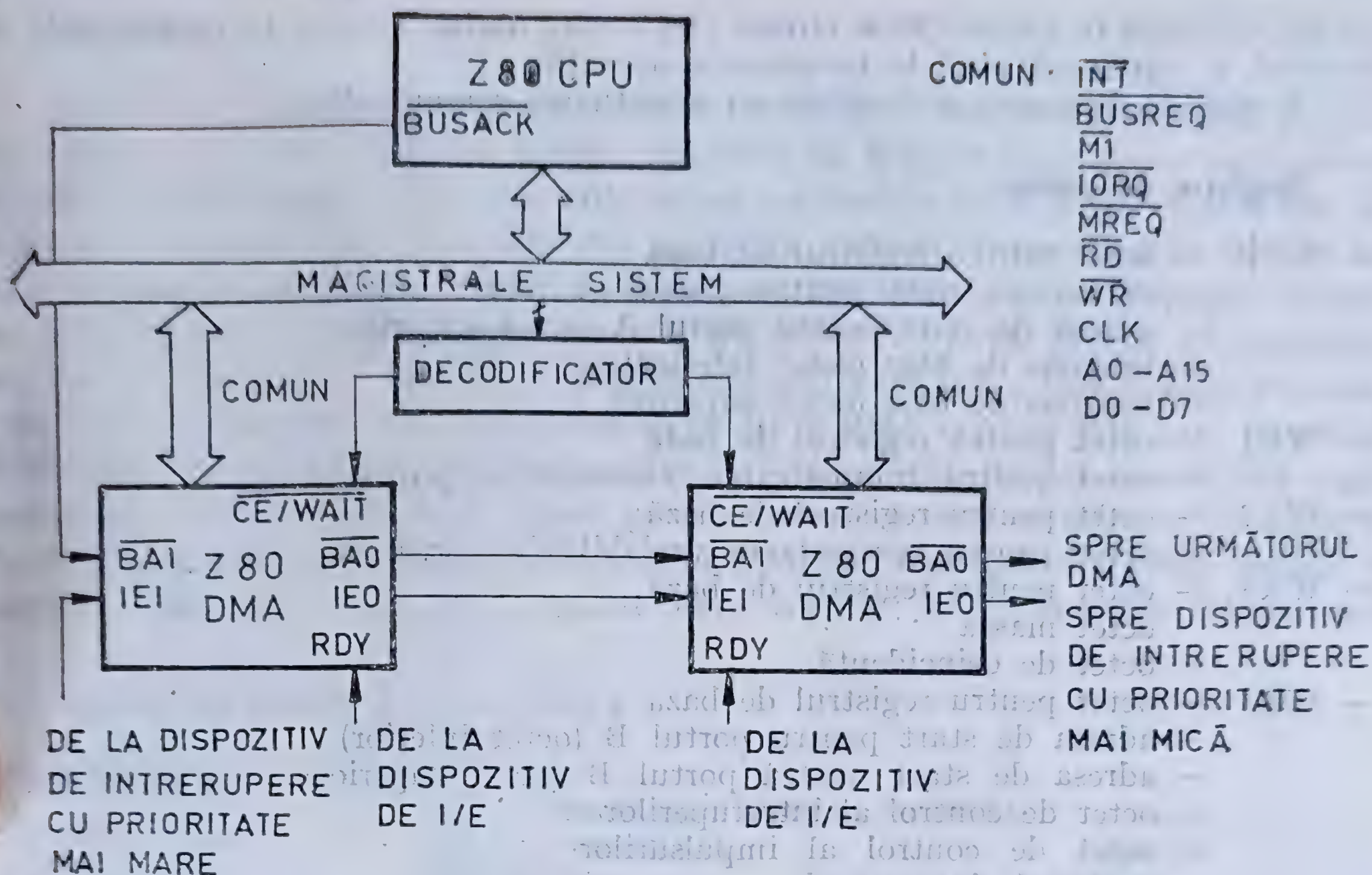


Fig. 7.6 Conectarea mai multor circuite Z80 DMA într-un sistem Z80.

În interiorul circuitului DMA există circuite logice specializate dedicate funcțiilor de interfațare de magistrală, control al magistralei interne, coincidență de octeți, generare de impulsuri periodice, întreruperi ale CPU, cereri de magistrală și generare de adrese. Un set de 21 de registre de control, care pot fi înscrise și 7 registre de stare, care pot fi citite fac posibil controlul unității centrale asupra funcționării acestor circuite. Toate registrele au capacitate de 8 biți, iar informațiile de 16 biți sînt depuse în registre adiacente. Cele două numărătoare de adresă (de câte 2 octeți fiecare), pentru portul A și B au ca tampon adresele de pornire. Cele 21 de registre de control sînt organizate în 7 grupuri de registre de bază, cele mai multe avînd registre multiple. Registrele de bază din fiecare grup conțin biți de control/comandă și biți indicatori care pot fi înscrise pentru a indica alte registre ale grupului. Cele 7 registre de stare nu au registre analoge pe nivelul 2.

Registrele sînt denumite în funcție de grupurile registrelor lor de bază:

- WR0—WR6 — grupurile de registre de scriere, de la 0 la 6 (7 registre de bază, plus 14 registre asociate)
- RR0—RR6 — registre de citire, de la 0 la 6.

Înscrierea unui registru dintr-un grup de registre de scriere implică înscrierea la început a registrului de bază, cu biții indicatori fixați corespunzător, apoi înscrierea unuia sau mai multor registre din grup. Toate cele 7 registre de stare (RR0—RR6) sînt accesibile secvențial, în conformitate cu o mască programabilă, conținută într-unul din registrele de scriere.

Pentru citirea datelor este utilizată o schemă tip conductă (pipe line). Lungimea programată a blocului este numărul de octeți comparat cu numărătorul de octeți, care este incrementat la sfîrșitul fiecărui ciclu. În operații de căutare, comparațiile de octeți cu octetul de coincidență sînt efectuate în timpul ciclului de citire al următorului octet. Coincidențele sînt, ca urmare, descoperite numai după citirea următorului octet.

În configurații cu mai multe circuite DMA, lanțul de priorități la întreruperi depinde de ordinea conectării liniilor IEI și IEO. Magistrala sistemului nu poate fi



totuși, eliberată în avans. Orice circuit DMA care obține accesul la magistralele sistemului, îl păstrează pînă la terminarea operației.

Registrele de scriere și de citire au următoarea semnificație:

### Registre de scriere

- WR0 — octet pentru registrul de bază
  - adresă de start pentru portul A (octet inferior)
  - adresă de start pentru portul A (octet superior)
  - lungime de bloc (octet inferior)
  - lungime de bloc (octet superior)
- WR1 — octet pentru registrul de bază
  - octet pentru temporizarea variabilă la portul A
- WR2 — octet pentru registrul de bază
  - octet pentru temporizare variabilă la portul B
- WR3 — octet pentru registrul de bază
  - octet mască
  - octet de coincidență
- WR4 — octet pentru registrul de bază
  - adresa de start pentru portul B (octet inferior)
  - adresa de start pentru portul B (octet superior)
  - octet de control al întreruperilor
  - octet de control al impulsurilor
  - vector de întreruperi
- WR5 — octet pentru registrul de bază
- WR6 — octet pentru registrul de bază
  - mască de citire

### Registre de citire

- RR0 — octet de stare
- RR1 — numărator de octeți (octet inferior)
- RR2 — numărator de octeți (octet superior)
- RR3 — numărator de adresă pentru portul A (octet inferior)
- RR4 — numărator de adresă pentru portul A (octet superior)
- RR5 — numărator de adresă pentru portul B (octet inferior)
- RR6 — numărator de adresă pentru portul B (octet superior)

### Programarea

Circuitul Z80 DMA are două stări fundamentale, programabile: prima, o stare activă, în care poate obține controlul magistrelor sistemului și poate direcționa transferul de date între două porturi; a doua, o stare inactivă, în care nu poate solicita magistralele și nu poate efectua transferuri de date. Cînd circuitul DMA este conectat la tensiunea de alimentare sau cînd este inițializat, este automat plasat în starea inactivă. Comenzile de programare de la unitatea centrală pot fi înscrise în circuitul DMA în oricare dintre stări, iar aceasta face ca circuitul DMA să treacă automat în stare inactivă, care este menținută pînă cînd unitatea centrală emite o comandă de activare. Unitatea centrală trebuie să programeze circuitul DMA înaintea oricărui transfer sau căutări de date, adresîndu-l ca port de I/E și trimițînd o secvență de octeți de control cu ajutorul unor instrucțiuni de ieșire (de exemplu OTIR).



## Înserierea

Octeții de control sau de comandă sînt înscrși într-unul sau mai multe grupuri de registre de scriere (WR0—WR6), înscrind la început octetul pentru registrul de bază al acelu grup. Toate grupurile au un registru de bază și cele mai multe au și registre adiționale asociate, care sînt accesibile secvențial, înscrind la început un octet în registrul de bază, pentru identificarea grupului de registre și pentru indicarea unuia sau mai multor registre asociate registrului de bază. Figura 7.7 ilustrează ordinea în care pot fi înscrise registrele asociate unui grup, prin poziția verticală a lor. Dacă de exemplu, un octet înscrș în circuitul DMA conține biții care identifică WR0 (D0, D1 și D7) și de asemenea conține „1” în pozițiile biților care indică registrul asociat pentru „Adresa de start pentru portul A, octetul inferior” și „Adresa de start pentru portul A, octetul superior”, atunci următorii doi octeți înscrși în circuitul DMA vor fi depuși în aceste două registre, în ordinea indicată mai sus.

### GRUPUL DE REGISTRE DE SCRIERE 0

D<sub>7</sub>D<sub>6</sub>D<sub>5</sub>D<sub>4</sub>D<sub>3</sub>D<sub>2</sub>D<sub>1</sub>D<sub>0</sub>

| 0

| | | | 0 0

| | | | 0 1

| | | | 1 0

| | | | 1 1

| | | | 0

| | | | 1

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

OCTET PENTRU REGISTRUL DE BAZA

NU SE UTILIZEAZA

TRANSFER

CAUTARE

CAUTARE/TRANSFER

PORT B → PORT A

PORT A → PORT B

ADRESA DE START PENTRU PORTUL A (OCTET INFERIOR)

ADRESA DE START PENTRU PORTUL A (OCTET SUPERIOR)

LUNGIME DE BLOC (OCTET INFERIOR)

LUNGIME DE BLOC (OCTET SUPERIOR)

### GRUPUL DE REGISTRE DE SCRIERE 1

D<sub>7</sub> D<sub>6</sub>

| 0 1 0 0

| | | | 0

| | | | 1

| | | | 0 0

| | | | 0 1

| | | | 1 0

| | | | 1 1

| | | | 0 0

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

| | | |

OCTET PENTRU REGISTRUL DE BAZA

PORTUL A ESTE MEMORIE

PORTUL A ESTE DI/E

ADRESA PORTULUI A SE DECREMENTEAZA

ADRESA PORTULUI A SE INCREMENTEAZA

ADRESA PORTULUI A ESTE FIXA

OCTET DE TEMPORIZARE VARIABILA PENTRU PORTUL A

LUNGIMEA CICLULUI DE 4T

LUNGIMEA CICLULUI DE 3T

LUNGIMEA CICLULUI DE 2T

NU SE UTILIZEAZA

$\overline{IORQ}$  SE TERMINA CU 1/2 CICLU IN AVANS

$\overline{MREQ}$  SE TERMINA CU 1/2 CICLU IN AVANS

$\overline{RD}$  SE TERMINA CU 1/2 CICLU IN AVANS

$\overline{WR}$  SE TERMINA CU 1/2 CICLU IN AVANS

Fig. 7.7 partea I



### GRUPUL DE REGISTRE DE SCRIERE 2

D <sub>7</sub>	D <sub>6</sub>
0	0 0 0

0  
1  
0 0  
0 1  
1 0  
1 1

OCTET PENTRU REGISTRUL DE BAZA  
PORTUL B ESTE MEMORIA  
PORTUL B ESTE DI/E  
ADRESA PORTULUI B SE DECREMENTEAZA  
ADRESA PORTULUI B SE INCREMENTEAZA  
ADRESA PORTULUI B, FIXA

0 0
0 1
1 0
1 1

OCTET DE TEMPORIZARE VARIABILA PENTRU PORTUL B  
LUNGIMEA CICLULUI DE 4T  
LUNGIMEA CICLULUI DE 3T  
LUNGIMEA CICLULUI DE 2T  
NU SE UTILIZEAZA

IORQ SE TERMINA CU 1/2 CICLU IN AVANS

MREQ

RD

WR

### GRUPUL DE REGISTRE DE SCRIERE 3

D <sub>7</sub>	D <sub>6</sub>
1	0 0

VALIDARE DMA  
VALIDARE INTRE-  
RUPERI

1 1

OCTET PENTRU REGISTRUL DE BAZA  
OPRIRE LA COINCIDENTA


OCTET MASCA (0=COMPARA)  
OCTET DE COINCIDENTA

### GRUPUL DE REGISTRE DE SCRIERE 4

D <sub>7</sub>	D <sub>6</sub>
1	0 1

0 0  
0 1  
1 0  
1 1

OCTET PENTRU REGISTRUL DE BAZA  
OCTET  
CONTINUU  
CONDITIONAT  
NU SE UTILIZEAZA


ADRESA DE START PENTRU PORTUL B  
(OCTET INFERIOR)  
ADRESA DE START PENTRU PORTUL B  
(OCTET SUPERIOR)

0
1 1

OCTET DE CONTROL AL INTRERUPERILOR  
INTRERUPERE LA COINCIDENTA  
INTRERUPERE LA SFIRSIT DE BLOC

INTRERUPERE  
LA RDY  
STAREA AFEC-  
TEAZA VECTORUL

1 1

SE GENEAREAZA UN IMPULS  
OCTET DE CONTROL AL IMPULSURILOR


VECTOR DE INTRERUPERI

0 0  
0 1  
1 0  
1 1

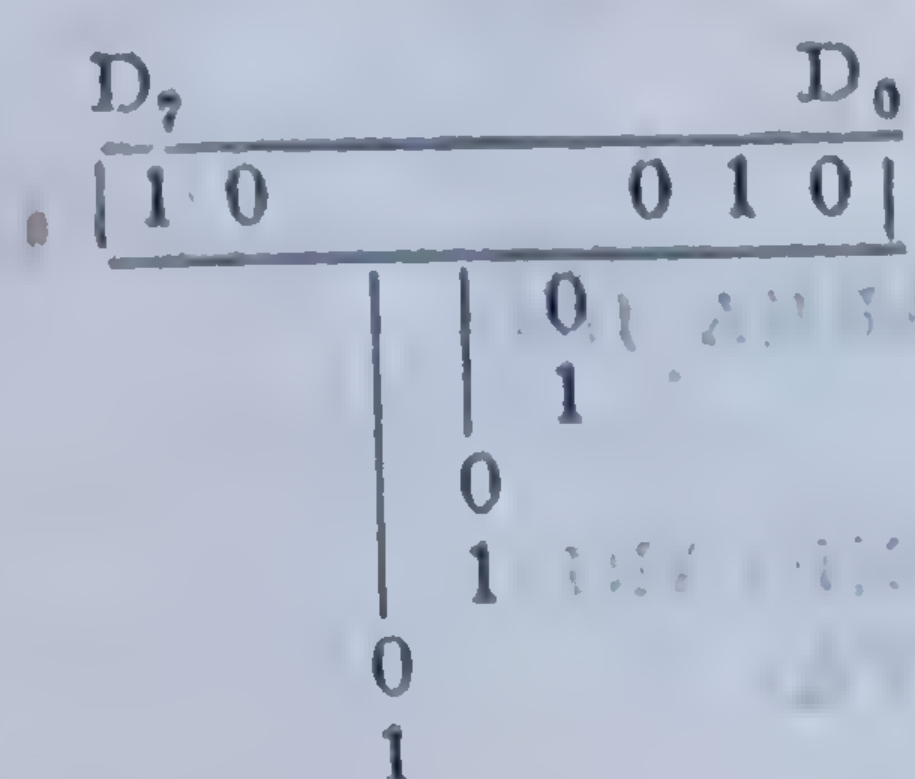
INTRERUPERE LA DRY  
INTRERUPERE LA COINCIDENTA  
INTRERUPERE LA SFIRSIT DE BLOC  
INTRERUPERE LA COINCIDENTA SI SFIRSIT  
DE BLOC

\* VECTORUL ESTE AUTOMAT MODIFICAT ASA CUM SE ARATA NUMAI DACĂ BITUL "STAREA AFECTEAZĂ VECTORUL" ESTE INSCRIS

Fig. 7.7. partea II (continuare)



## GRUPUL DE REGISTRE DE SCRIERE 5



OCTET PENTRU REGISTRUL DE BAZA  
 READY ESTE ACTIV PE 0  
 READY ESTE ACTIV PE 1  
 NUMAI CE  
 CE/WAIT, MULTIPLEXATE  
 STOP LA SFIRSITUL BLOCULUI  
 AUTORESTART LA SFIRSIT DE BLOC

## GRUPUL DE REGISTRE DE SCRIERE 5



HEXAZECIMAL

1 1 0 0 0 0 1 1  
 1 1 0 0 0 1 1 1  
 1 1 0 0 1 0 1 1  
 1 1 0 0 1 1 1 1  
 1 1 0 1 0 0 1 1  
 1 0 1 0 1 1 1 1  
 1 0 1 0 1 0 1 1  
 1 0 1 0 0 0 1 1  
 1 0 1 1 0 1 1 1  
 1 0 1 1 1 1 1 1  
 1 0 0 0 1 0 1 1  
 1 0 1 0 0 1 1 1  
 1 0 1 1 0 0 1 1  
 1 0 0 0 0 1 1 1  
 1 0 0 0 0 0 1 1  
 1 0 1 1 1 0 1 1

C3 INITIALIZARE  
 C7 INITIALIZARE, TIMING PORT A  
 CB " PORT B  
 CF INCARCA  
 D3 CONTINUA  
 AF DEZACTIVEAZA INTRERUPERILE  
 AB ACTIVEAZA INTRERUPERILE  
 A3 INITIALIZARE SI DEZACTIVARE INTRERUPERI  
 B7 VALIDARE DUPA RETI  
 BF CITESTE OCTET DE STARE  
 8B REINITIALIZEAZA OCTET DE STARE  
 A7 INCEPE SECVENTA DE CITIRE  
 B3 FORTEAZA READY  
 87 ACTIVEAZA DMA  
 83 DEZACTIVEAZA DMA  
 BB URMEAZA MASCA DE CITIRE



MASCA DE CITIRE (1=VALIDEAZA)

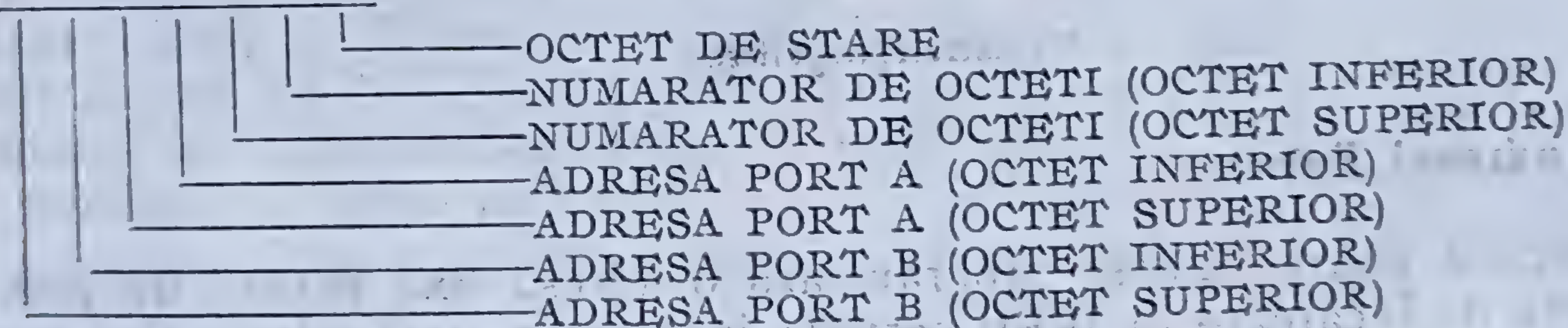


Fig. 7.7 Registrele de scriere.

### Citirea

Registrele de citire RR0—RR6 sînt citite de unitatea centrală adresînd circuitul DMA ca port de I/E, cu instrucțiuni de intrare (de exemplu INIR). Octeții care pot fi citiți conțin starea circuitului DMA, valoarea număratorului de octeți, și adresele de port de la ultima inițializare a circuitului DMA.

Registrele se citesc întotdeauna într-o secvență fixată, începînd cu RR0 și terminînd cu RR6. Totuși, registrul citit în această secvență este determinat de programarea octetului mască de citire în WR6. Secvența de citire este inițializată înscrind în WR6 o comandă „Începe secvența de citire” sau „Fixează starea de citire”. După o inițializare de DMA, secvența trebuie inițializată cu o comandă „Începe secvența de citire” sau „Citește starea”.

Secvența de citire a tuturor registrelor care nu sînt excluse de octetul mască de citire, trebuie terminată înaintea unei noi comenzi „Începe secvența de citire” sau „Citește starea”. Registrele de citire sînt reprezentate în figura 7.8.



NAREA DMA  
STEPTARE  
ENTA

TETI (OCTET INFERIOR)

CTETI (OCTET SUPERIOR)

RESA PENTRU PORTUL A (OCTET INFE

DRESA PENTRU PORTUL A (OCTE

DRESA PENTRU PORTUL B (OCTET

ADRESA PENTRU PORTUL B (OCTE  
UNO MIL NOUĂ

Fig. 7.8 Registrele de citire.

amează o adresă fixă pentru un port, care permite încărcarea unei adrese fixe în registrul de destinație. De aceea, o adresă fixă pentru un moment ca adresă fixă pentru un port, ce o face implicit pe prima, să o considerăm ca o astfel din această procedură, presupunând o sursă cu adresă variabilă (portul A este variabil în timp).

6.  $\text{DMA} \rightarrow \text{WR6}$

transfer de date din memorie (por  
acest exemplu, adresa de start a  
pozitiv de I/I are adresa fixă 05<sub>H</sub>  
l, deci cu unul mai mult decât l  
u circuitul DMA poate fi format în  
circuitul DMA cu o instrucțiune de



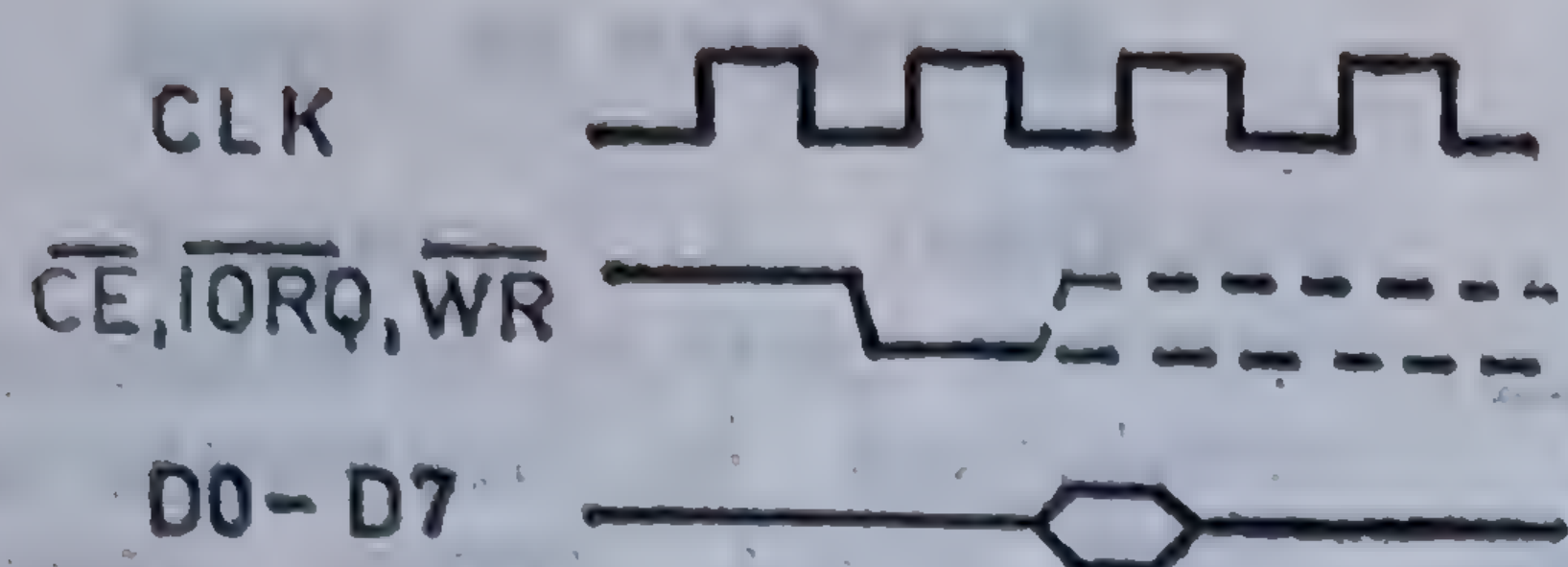


Fig. 7.9 Ciclu de scriere CPU  $\rightarrow$  DMA.

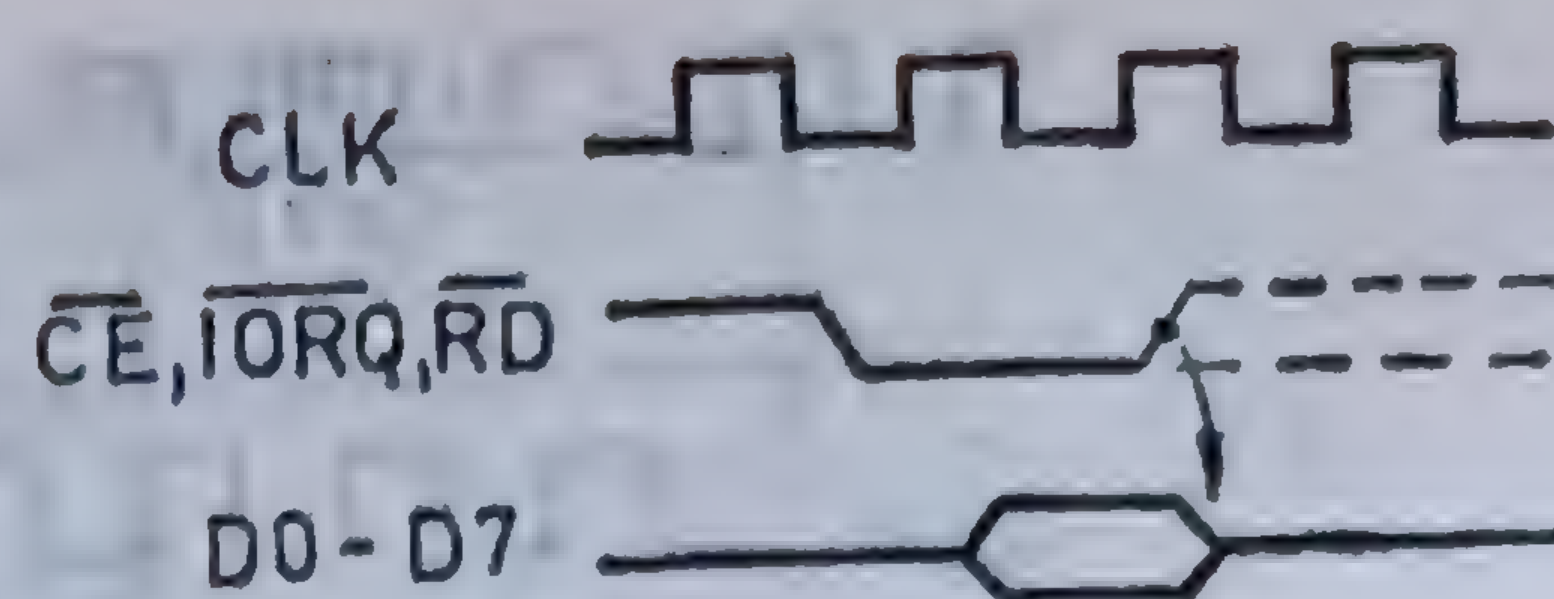


Fig. 7.10 Ciclu de citire CPU  $\rightarrow$  DMA.

### Funcționarea în timp a circuitului DMA

#### Funcționarea în timp în starea inactivă, ca periferic al CPU.

În starea dezactivată (inactivă) circuitul DMA este adresat de CPU ca un periferic de I/E, pentru operații de scriere sau citire (pentru control, respectiv stare).

Diagrama în timp a unei operații de scriere este prezentată în figura 7.9. Citirea octetului de stare al circuitului DMA, număratorului de octeți, sau numărătoarelor adreselor de port este ilustrată în figura 7.10. Aceste operații necesită mai puțin de 3 cicluri de tact. Liniile  $\overline{CE}$ ,  $\overline{IORQ}$  și  $\overline{RD}$  devin active la două fronturi crescătoare ale semnalului CLK și datele apar pe magistrală aproximativ la o perioadă de tact după ce ele devin active.

#### Funcționarea în timp în starea activă ca și controler de magistrale

Implicit, și după inițializare (RESET), desfășurarea în timp a operațiilor de citire și de scriere este aceeași cu a ciclurilor de citire și de scriere pentru memorie și dispozitive de I/E, cu o excepție: în timpul unui ciclu de citire, datele sînt captate pe frontul căzător al lui  $T_3$  și menținute pe magistrala de date peste limita dintre ciclurile de citire și de scriere, pînă la trecerea următorului ciclu de citire. Figura 7.11 ilustrează desfășurarea în timp a unui transfer de la memorie spre un port de I/E iar figura 7.12 ilustrează transferul în sens invers. Transferurile memorie—memorie și port de I/E—port de I/E sînt simple permutări ale acestor diagrame. Desfășurarea în timp în cazul implicit utilizează trei cicluri T de tact pentru operații cu memoria și patru perioade de tact pentru operații de I/E, care includ un ciclu de așteptare, inserat automat între  $T_2$  și  $T_3$ . Dacă linia  $\overline{CE}/\overline{WAIT}$  este programată

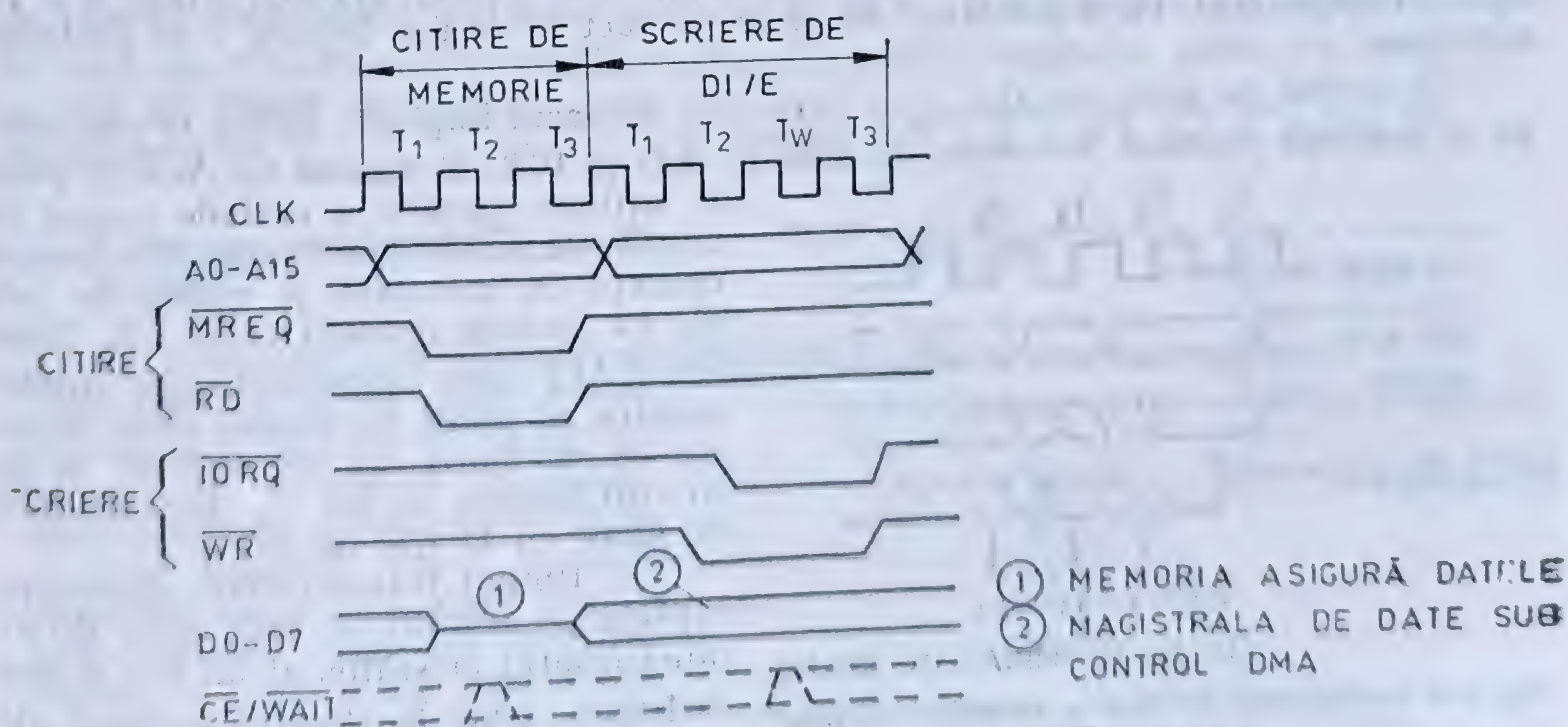


Fig. 7.11 Transfer de la memorie spre DI/E.



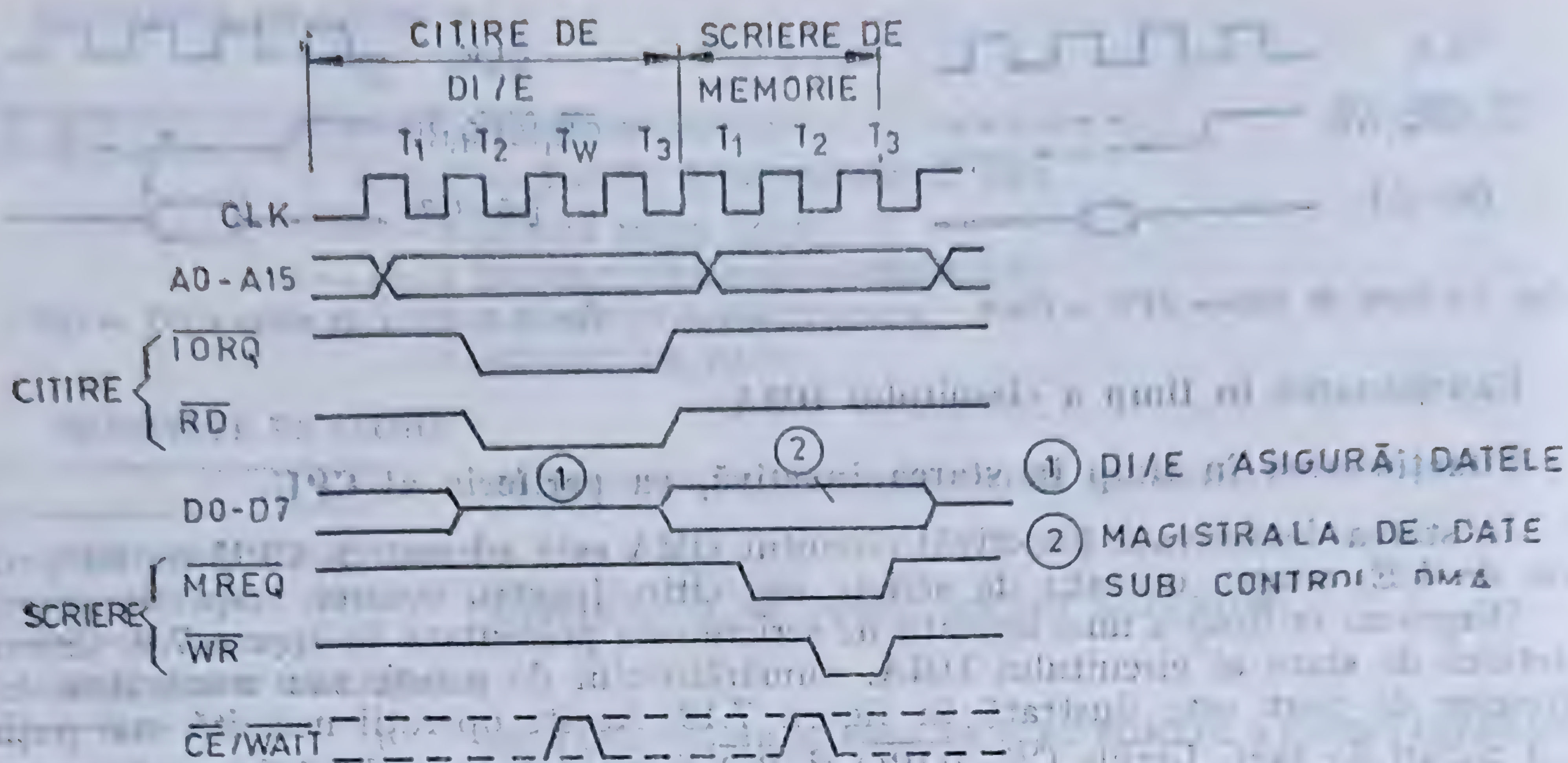


Fig. 7.12 Transfer de la DI/E spre memorie.

să funcționeze ca linie  $\overline{\text{WAIT}}$  în timpul stării active a circuitului DMA, ea este eșantionată pe frontul căzător al lui  $T_2$  pentru operații cu memoria și pe frontul căzător al lui  $T_w$  pentru operații de I/E. Dacă  $\overline{\text{CE/WAIT}}$  este la 0 în acest moment, este adăugat un alt ciclu  $T$  în timpul căruia linia  $\overline{\text{CE/WAIT}}$  este din nou eșantionată. Durata operațiilor poate fi astfel extinsă nedefinit.

### Desfășurarea în timp a ciclurilor variabile și a fronturilor

Lungimea ciclului de funcționare implicit pentru circuitul Z80 DMA la portul sursă (de citire) poate fi programată independent de cea de la portul de destinație (de scriere). Caracteristica de ciclu variabil permite cicluri de citire sau de scriere de 2, 3 sau 4 cicluri de tact  $T$  (sau mai multe, dacă se inserază cicluri de așteptare), crescând sau scăzând astfel viteza tuturor semnalelor generate de circuitul DMA.

În plus, fronturile de terminare ale semnalelor  $\overline{\text{IORQ}}$ ,  $\overline{\text{MREQ}}$ ,  $\overline{\text{RD}}$  și  $\overline{\text{WR}}$  pot apare, independent, cu o jumătate de ciclu de tact în avans. Figura 7.13 ilustrează acest caz.

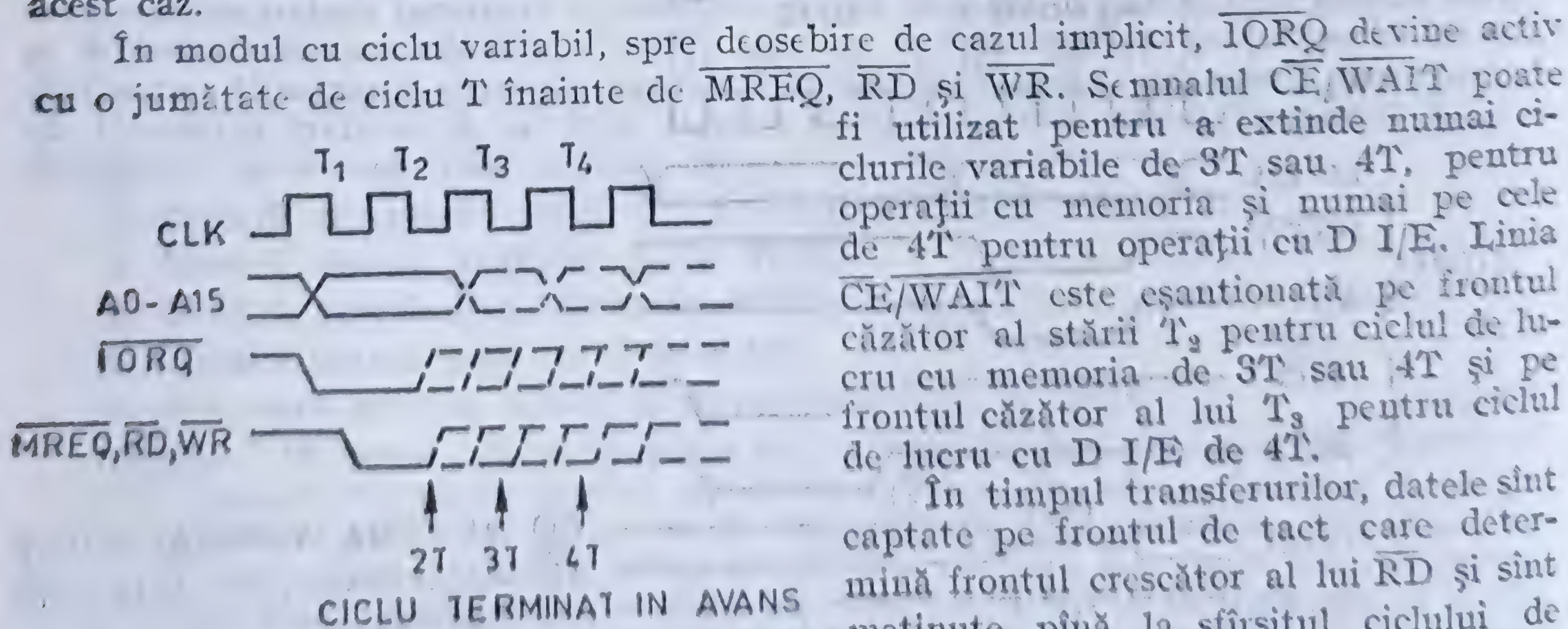


Fig. 7.13 Desfășurarea în timp a ciclurilor variabile și a fronturilor.

În timpul transferurilor, datele sînt captate pe frontul de tact care determină frontul crescător al lui  $\overline{\text{RD}}$  și sînt menținute pînă la sfîrșitul ciclului de scriere inclusiv.



### Cereri de magistrală

Figura 7.14 ilustrează desfășurarea în timp a cererii de magistrală și acceptarea acesteia. Linia RDY, care poate fi programată să fie activă pe 1 sau pe 0, este eșantionată pe fiecare front crescător al semnalului CLK. Dacă este găsită activă și dacă magistrala nu este utilizată de un alt dispozitiv, următorul front crescător al lui CLK aduce linia BUSREQ la 0. După recepționarea acestei cereri de magistrală, unitatea centrală anunță acceptarea ei pe linia de intrare BAI a circuitului DMA,

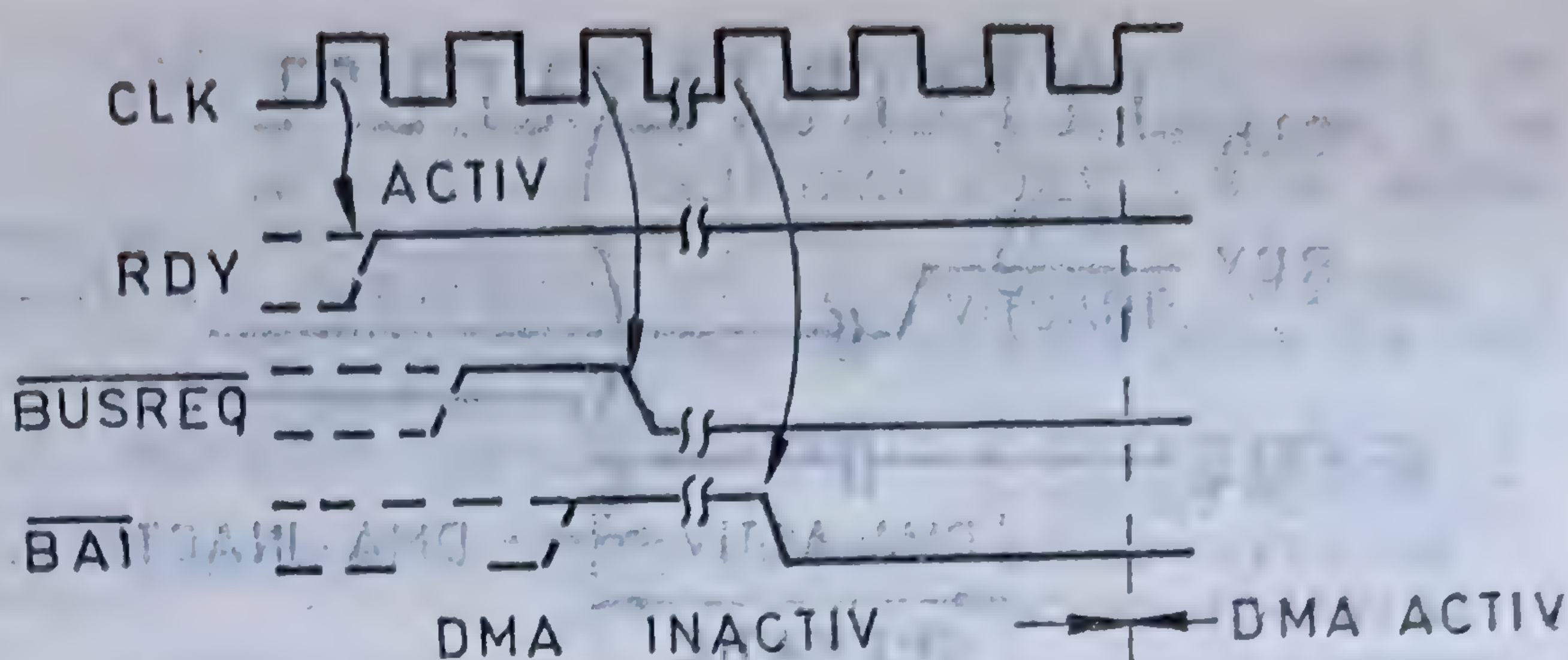


Fig. 7.14 Cerere de magistrală și acceptarea ei.

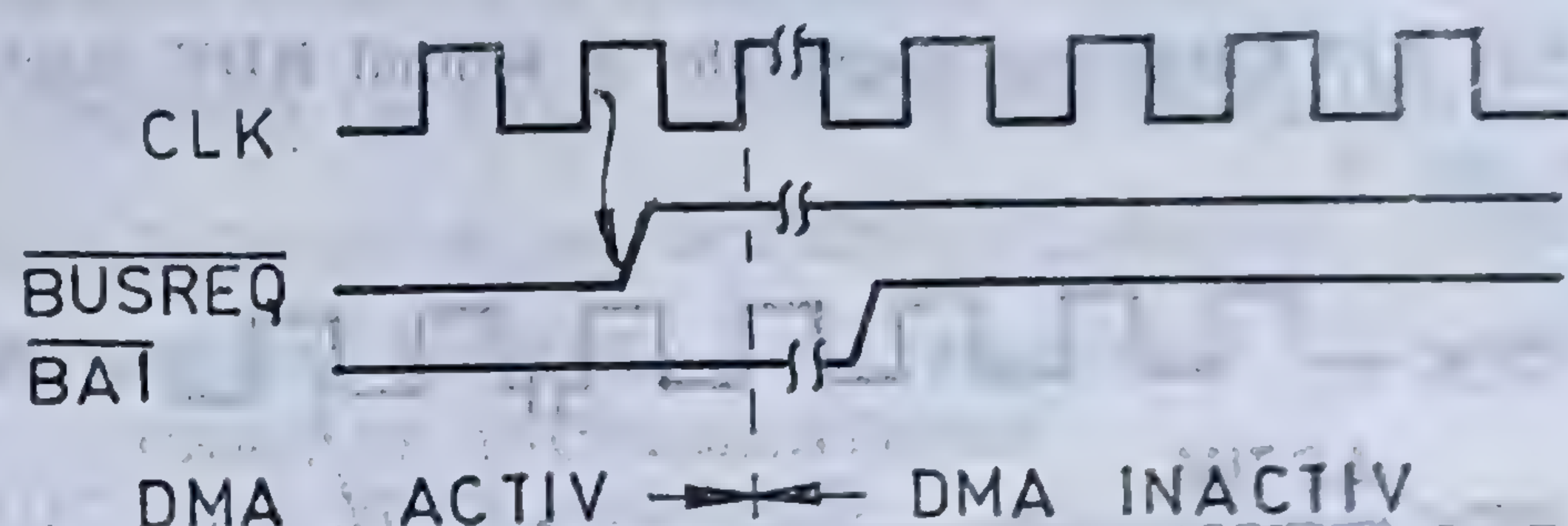


Fig. 7.15 Eliberarea magistralei (modul „cîte un octet”).

direct, sau printr-un lanț de priorități DMA. Când linia BAI este la 0 timp de 2 fronturi crescătoare consecutive ale semnalului CLK, circuitului DMA va începe transferul de date pe următorul front crescător al lui CLK.

### Eliberarea magistralei în modul „cîte un octet”

În modul de transfer „cîte un octet”, linia BUSREQ este adusă la 1 pe frontul crescător al lui CLK, care precede sfîrșitul fiecărui ciclu de citire (în cazul căutării fără transfer) sau sfîrșitul fiecărui ciclu de scriere (în cazul operațiilor de transfer sau de transfer și căutare), așa cum se ilustrează în figura 7.15. Aceasta are loc indiferent de starea liniei RDY. Nu există nici o posibilitate de confuzie cînd se utilizează unitatea centrală Z80 CPU, care începe să funcționeze în următorul ciclu de tact T. Majoritatea unităților centrale de alt tip funcționează de asemenea corect în acest caz. Următoarea cerere de magistrală, pentru următorul octet, va apare după ce atît BUSREQ cît și BAI vor reveni la 1.

### Eliberarea magistralei la sfîrșit de bloc

În modulele „condiționat” și „continuu”, apariția unui sfîrșit de bloc face ca linia BUSREQ să treacă la 1, de obicei pe același front crescător al semnalului CLK pe care circuitul DMA termină transferul blocului de date (figura 7.16). Ultimul octet din bloc este transferat, chiar dacă linia RDY devine inactivă înaintea terminării transferului ultimului octet.

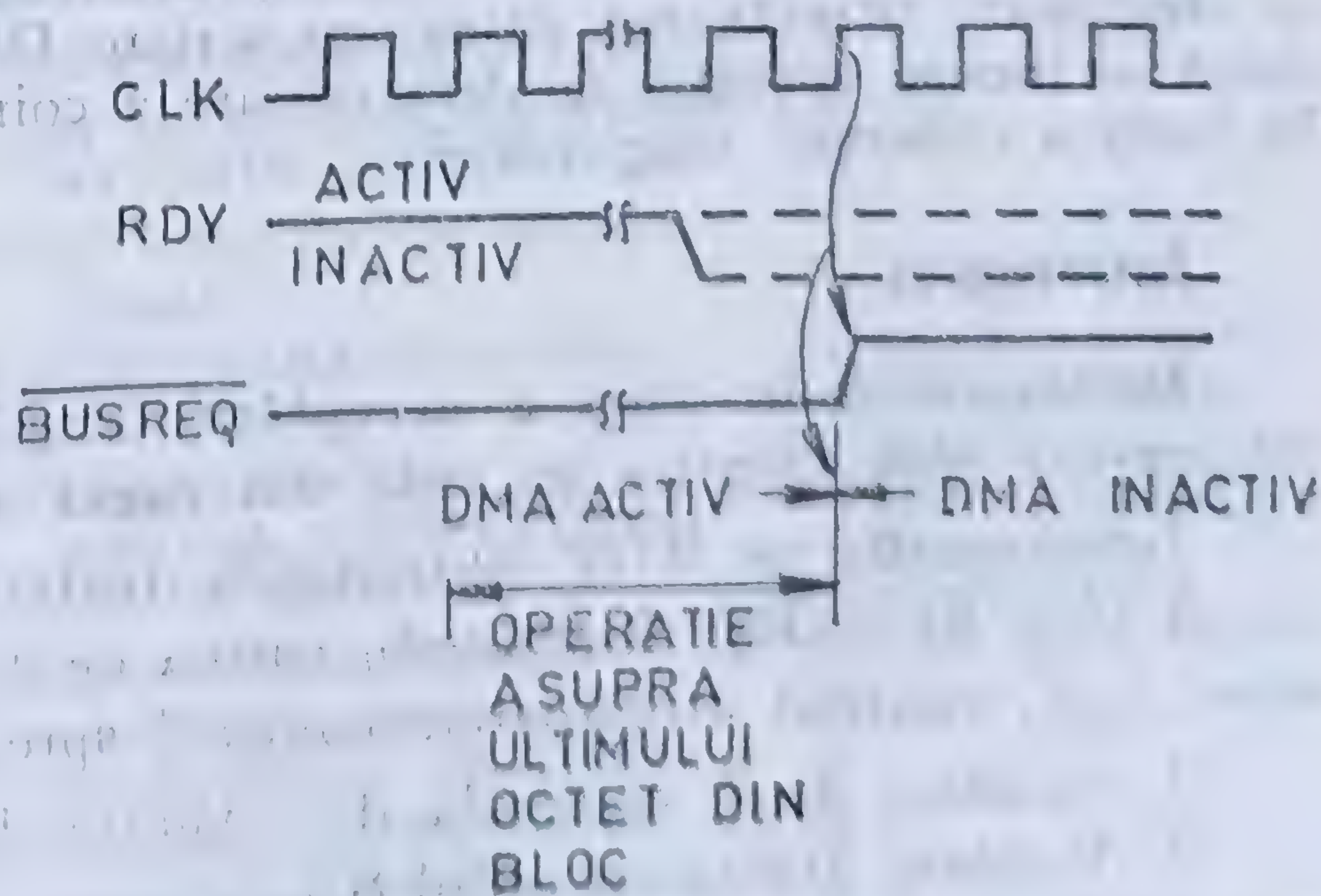


Fig. 7.16 Eliberarea magistralei la „sfîrșit de bloc” (modul „condiționat” sau „continuu”).



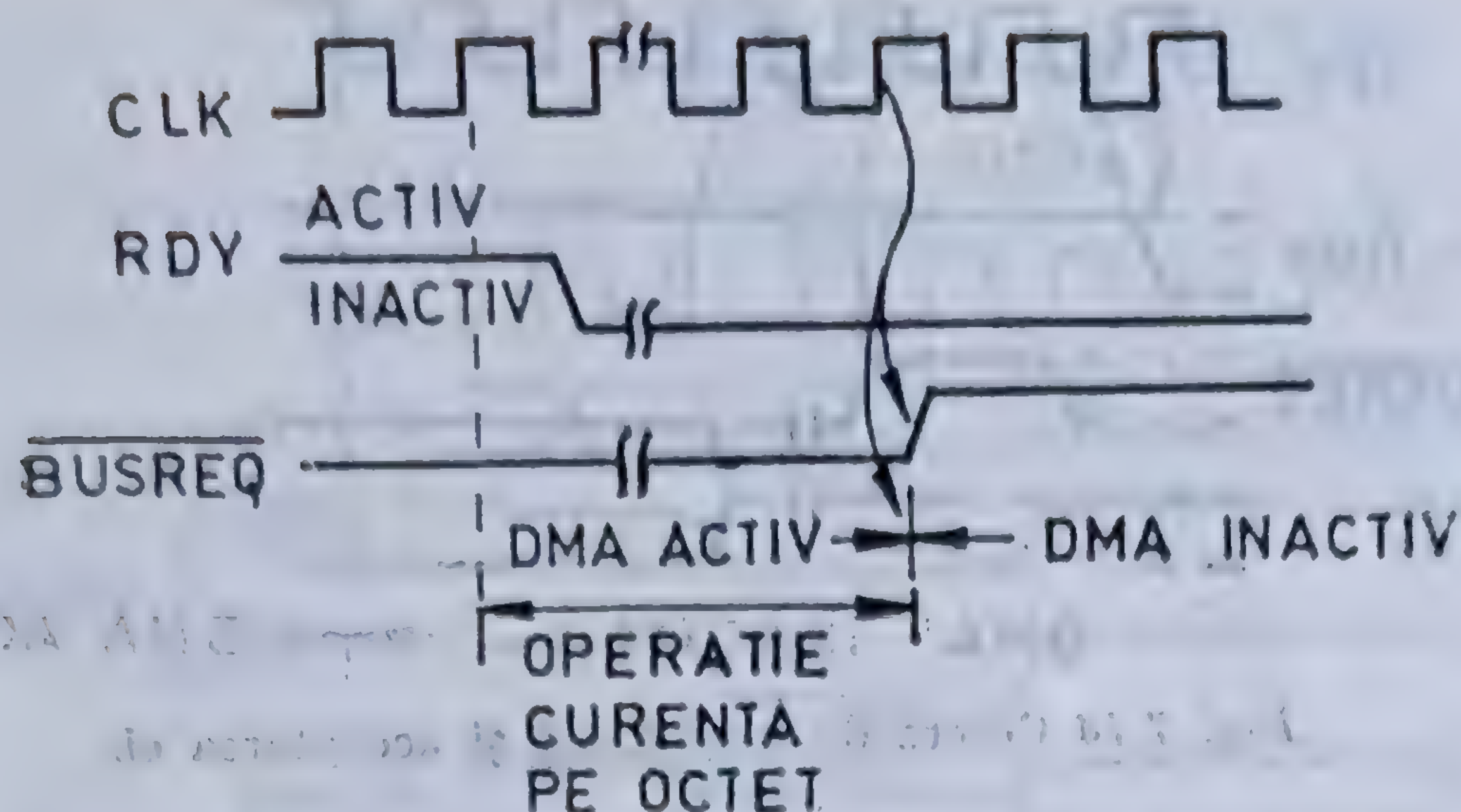


Fig. 7.17 Eliberarea magistralei la semnal RDY inactiv.

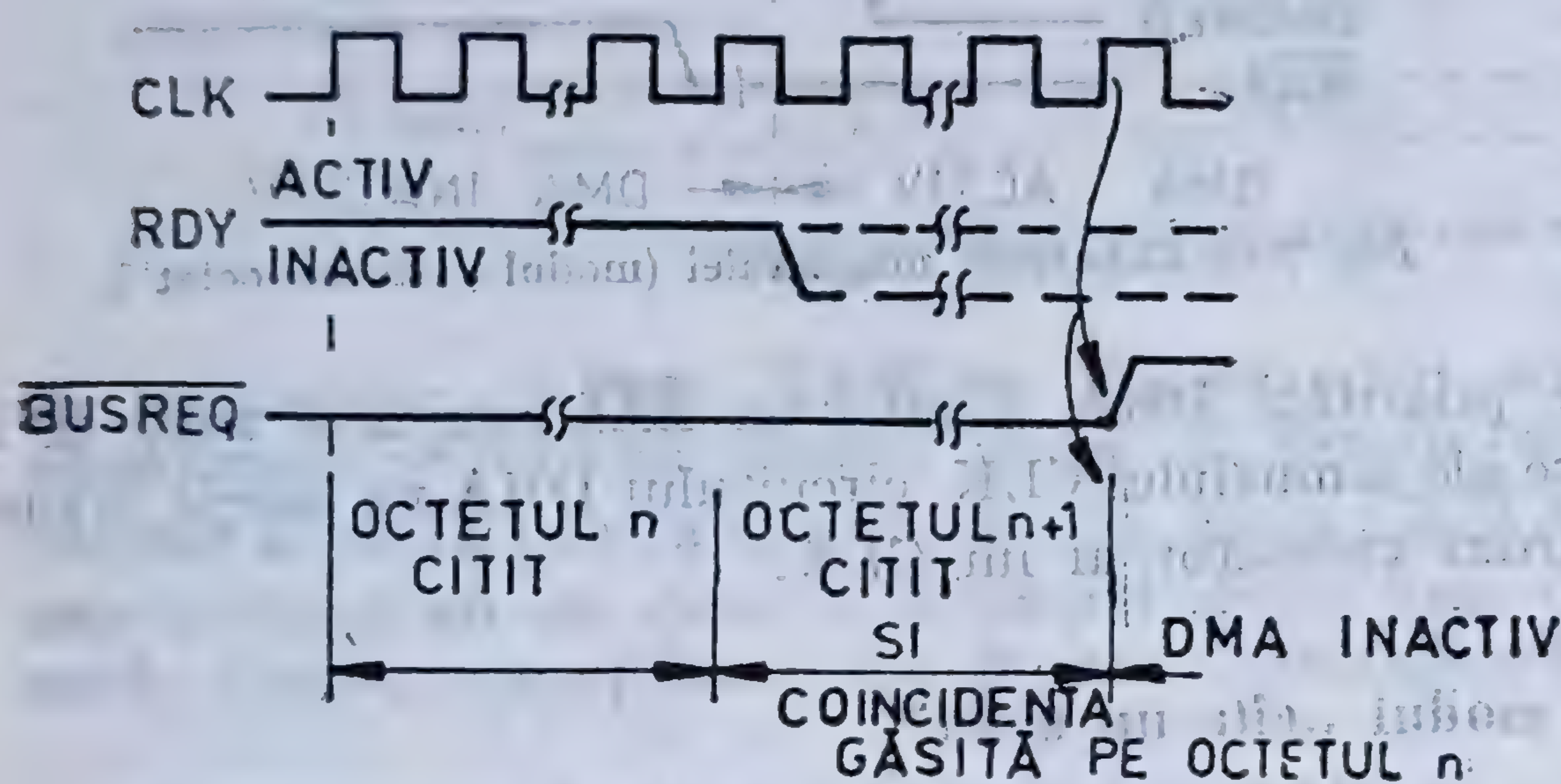


Fig. 7.18 Eliberarea magistralei la coincidență (modurile „condiționat” sau „continuu”).

### Eliberarea magistralei la apariția unei coincidențe

Dacă circuitul DMA este programat să se oprească la găsirea unei coincidențe în modurile „condiționat” sau „continuu”, apariția ei determină linia BUSREQ să devină inactivă pe următoarea operație DMA, deci la sfârșitul următoarei citiri într-o operație de căutare, sau la sfârșitul următoarei scrieri, într-o operație de transfer (figura 7.18). Datorită schemei de tip conductă, coincidențele sînt găsite în timp ce se efectuează următoarea citire sau scriere DMA. Linia RDY poate deveni inactivă după ce începe operația de determinare a coincidenței, fără să afecteze desfășurarea în timp a eliberării magistralei.

### Întreruperi

Desfășurarea în timp a acceptării cererilor de întrerupere și a revenirilor din întrerupere sînt identice cu cele din cazul altor periferice ale familiei Z80.

Întreruperile pe RDY (întrerupere înaintea cererii de magistrală) nu afectează direct linia BUSREQ. În schimb, rutina de servire a întreruperii trebuie să asigure acest fapt, emițînd următoarele comenzi spre WR6:

1. Validare după comanda de revenire din întrerupere (RETI), cu codul B7<sub>H</sub>.
2. Validare DMA, cu codul 87<sub>H</sub>.
3. O instrucțiune RETI care inițializează bistabilul de „Întrerupere în curs de servire” din circuitul DMA.

### Eliberarea magistralei la semnal READY inactiv

În modul „condiționat”, cînd linia RDY devine inactivă, ea determină linia BUSREQ să treacă la 1 pe următorul front crescător al lui CLK, după terminarea operației curente pe octet (figura 7.17). Acțiunea asupra liniei BUSREQ este astfel întîrziată într-o anumită măsură față de acțiunea asupra liniei RDY. Circuitul DMA termină întotdeauna operația curentă asupra unui octet, înainte de eliberarea magistralei.

Prin contrast, linia BUSREQ nu este eliberată, în modul „continuu”, dacă semnalul RDY devine inactiv. Circuitul DMA intră într-o stare latentă după terminarea operației curente asupra unui octet, așteptînd ca linia RDY să devină din nou activă.



WR0 79H	0 1 1 1 1 0 0 1	PREGATESTE DMA PT. PRIMIREA LUNGIMII DE BLOC, ADRESEI DE START PT. PORTUL A SI STABILESTE TEMPORAR PORTUL B CA SURSA
50H	0 1 0 1 0 0 0 0	ADRESA PORTULUI A, OCTET INFERIOR
10H	0 0 0 1 0 0 0 0	ADRESA PORTULUI A, OCTET SUPERIOR
00H	0 0 0 0 0 0 0 0	LUNGIMEA DE BLOC, OCTETUL INFERIOR
10H	0 0 0 1 0 0 0 0	LUNGIMEA DE BLOC, OCTETUL SUPERIOR
WR1 14H	0 0 0 1 0 1 0 0	DEFINESTE PORTUL A CA MEMORIE, CU ADRESA CARE SE INCREMENTEAZA
WR2 28H	0 0 1 0 1 0 0 0	DEFINESTE PORTUL B CA PERIFERIC, CU ADRESA FIXA
WR4 C5H	1 1 0 0 0 1 0 1	STABILESTE MODUL CONDITIONAT SI PREGATESTE CIRCUITUL DMA PENTRU PRIMIREA ADRESEI PORTULUI B
05H	0 0 0 0 0 1 0 1	ADRESA PORTULUI B (OCTETUL INFERIOR), 05H
WR5 8AH	1 0 0 0 1 0 1 0	STABILESTE NIVELUL ACTIV 1 PENTRU SEMNALUL READY
WR6 CFH	1 1 0 0 1 1 1 1	INCARCA ADRESA PORTULUI B SI INITIALIZEAZA NUMARATORUL DE BLOC
WR0 05H	0 0 0 0 0 1 0 1	STABILESTE PORTUL A CA SURSA
WR6 CFH	1 1 0 0 1 1 1 1	INCARCA ADRESA PORTULUI A SI INITIALIZEAZA NUMARATORUL DE BLOC
WR6 87H	1 0 0 0 0 1 1 1	VALIDEAZA CIRCUITUL DMA, PENTRU A INCEPE PUNCTIONARUA

ACESTI OCTETI SINT NECESARI NUMAI IN CAZUL UNEI ADRESE FIXE DE DESTINATIE

Fig. 7.19 Set de cuvinte de comandă pentru transferul unui bloc de date.



## 7.P2. ALICAȚII ALE CIRCUITULUI Z80 DMA

### 1. Transfer {do} date din memorie, la un port de I/E, cu adresă fixă

Se dă în continuare setul de cuvinte de comandă și control pentru transferul unui bloc de date de lungime  $1000_H$  din memorie, începînd de la adresa  $1050_H$ , la portul de I/E cu adresa  $05_H$  (figura 7.19).

Cei 14 octeți se înscriu în memorie la adresa pe care o notăm cu  $DMATBL$  ( $C000_H$ ) iar segmentul de program care realizează inițializarea circuitului DMA, a cărui adresă a portului de comandă este  $DMACOM = 10_H$ , și transferul de date, este următorul:

CODUL	ETICHETA	COD OPERAȚIE	COMENTARIU
2100C0	DMAINI:	LD HL,DMATBL	; ADRESA TABEL DE COMENZI, ÎN
			; HL
0E10		LD C,DMACOM	; ADRESA PORT DE COMANDA DMA
060E		LD B,LTBDMA	; LUNGIME TABEL COMENZI ( $14_D$ )
EDB3		OTIR	; INSCRIE CUVINTE DE COMANDA
C9		RET	; REVINE

Această subrutină se apelează din programul principal, pentru a efectua transferul dorit.

### 2. Căutarea unui octet în memorie

Cuvintele de comandă necesare căutării în memorie, începînd de la adresa  $5000_H$ , pe o lungime de bloc de  $1000_H$ , a unui octet al cărui semioctet superior este  $1001_B$  ( $5_H$ ) sînt cele din figura 7.20.

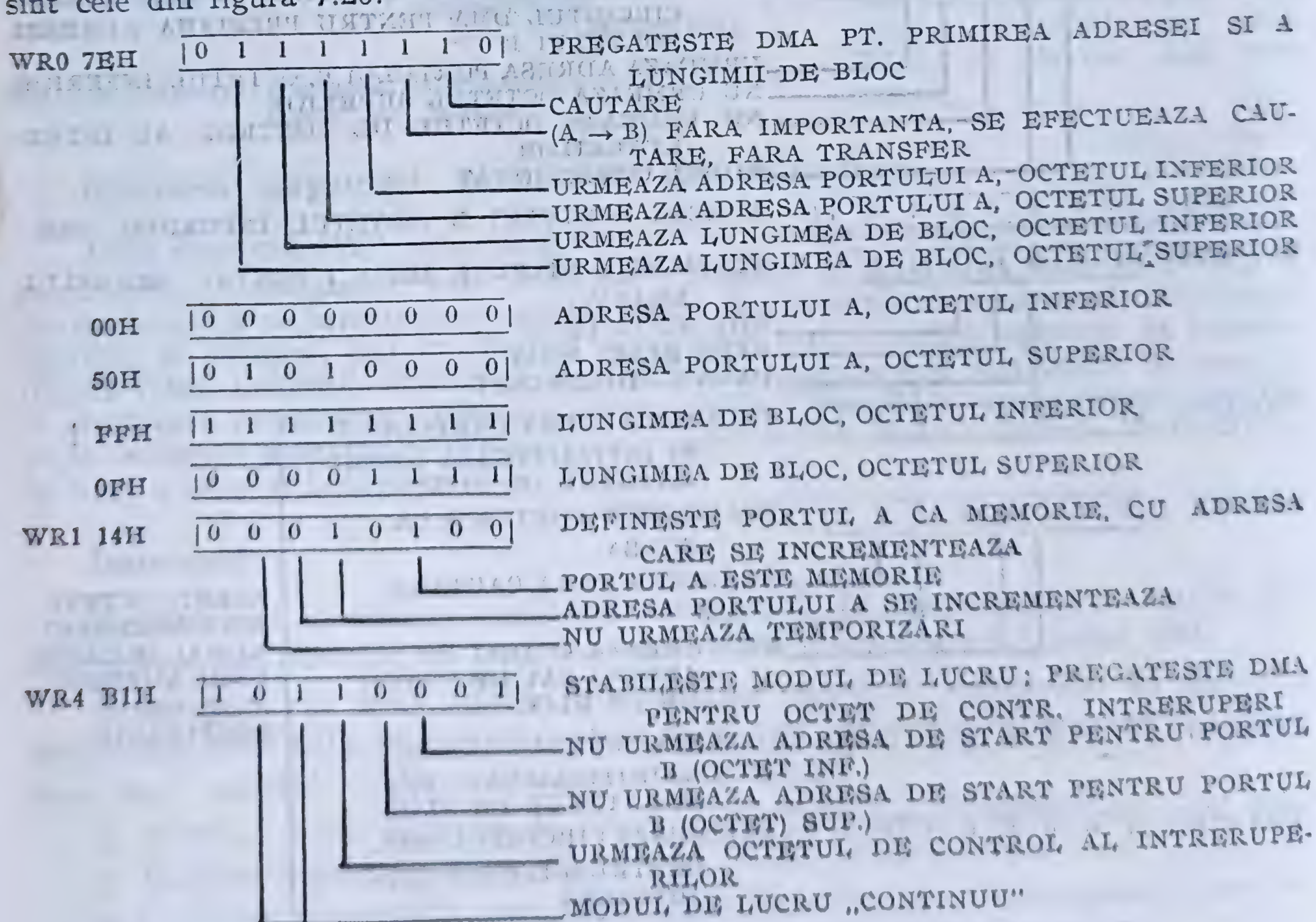


Fig. 7.20 (partea I)



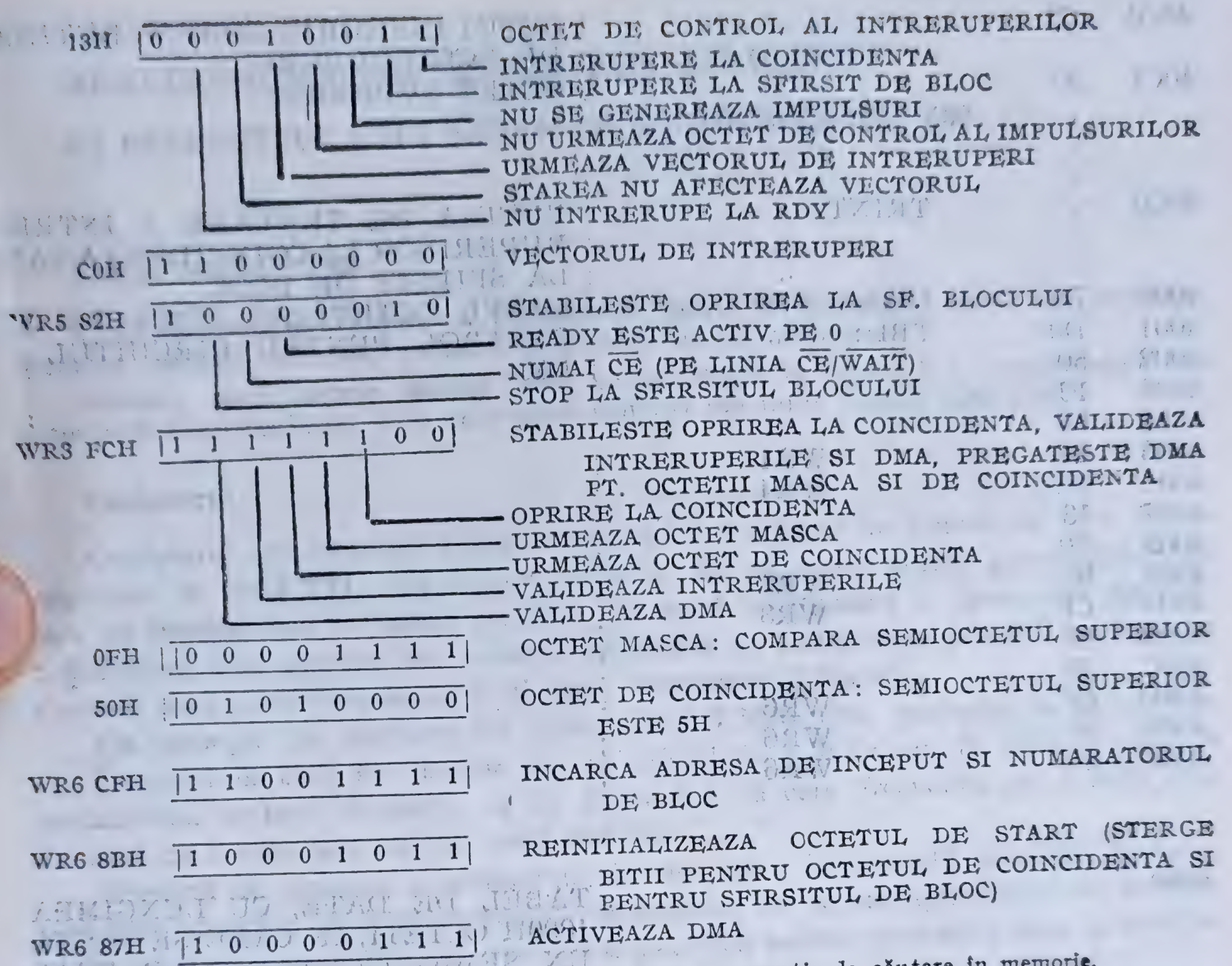


Fig. 7.20 Set de cuvinte de comandă pentru o operație de căutare în memorie.

Presupunem tabelul celor 16 octeți de comandă și control înscris în memorie la adresa DMATBL = 4000<sub>H</sub>, programul principal la adresa 2000<sub>H</sub>, iar subrutina de tratare a întreruperilor la 3000<sub>H</sub>, adresa ei fiind înscrisă la adresa 40C0<sub>H</sub>, dată de octetul din registrul I și de vectorul de întreruperi. Adresa portului DMA este DMACOM = 60<sub>H</sub>. Programul principal și tabelul cu octeții de comandă și control pentru circuitul DMA, sînt următoarele:

ADRE- SA	CODUL	ETI- CHETA	COD OPERATIE	COMENTARIU
2000	3E40	PRO- GPR:	LD A, 40H	; OCTET SUPERIOR AL ADRESEI ; TABELEI CU ADRESELE RUTINE- ; LOR DE INTRERUPERE
2002	ED47		LD I, A	; ADRESA TABEL DE COMENZI
2004	210040		LD HL, DMATBL	
2007	0E60		LD C, DMACOM	; ADRESA PORT DE COMANDA DMA
2009	0610		LD B, LTBDMA	; LUNGIME TABEL COMENZI (16 <sub>D</sub> )
200B	EDB3		OTIR	; INSCRIE CUVINTE DE COMANDA
200D	C9		RET	; REVINE



40C0 C0 ; OCTET INFERIOR ADRESA RUTINA  
DE INTRERUPERI ;  
40C1 30 ; OCTET SUPERIOR

30C0 . TRINT: ; RUTINA DE TRATARE A INTRE-  
RUPERILOR LA COINCIDENTA SAU  
LA SFIRSIT DE BLOC

4000 7E DMA-WR0 ; TABEL OCTETI DE COMANDA SI  
4001 D0 TBL: ; CONTROL PENTRU CIRCUITUL  
4002 50 DMA  
4003 FF

4004 0F

4005 14 WR1

4006 B1 WR4

4007 13

4008 C0

4009 82 WR5

400A CF WR3

400B 0F

400C 50

400D CF WR6

400E 83 WR6

400F 87 WR6

5000 . TABEL DE DATE, CU LUNGIMEA  
1000H OCTETI, IN CARE SE CAUTA  
UN SEMIOCTET SUPERIOR EGAL  
CU 5H

5FFF



## CAPITOLUL VIII

### REALIZAREA UNUI SISTEM Z80. APLICAȚII

#### 8.1 ELEMENTELE UNUI SISTEM CU MICROPROCESOR Z80

##### Sistem minimal Z80

Orice sistem Z80 trebuie să includă cel puțin 5 elemente: sursă de tensiune, oscilator, unitate centrală, memorii și circuite de intrare/ieșire.

Schema unui sistem minim este reprezentată în figura 8.1, comunicarea cu exteriorul fiind realizată prin cele două porturi ale unui circuit Z80 PIO.

##### Oscilatorul

Oscilatorul este în general foarte simplu, fiind necesar un semnal de tact dreptunghiular cu nivel TTL. Un simplu oscilator RC se poate utiliza pentru sistemele care nu funcționează la viteza maximă. În cazul funcționării în apropierea frecvenței maxime, este necesar un oscilator cu cristal de cuarț, care poate fi realizat cu ajutorul unor porți inversoare și al unor componente discrete.

Un exemplu de oscilator cu cristal de 2,5 MHz este prezentat în figura 8.2.

În cazul în care se dispune de un cristal de 5 MHz sau 10 MHz, frecvența oscilatorului trebuie divizată, ca în figura 8.3, în care frecvența de 5 MHz este divizată cu 2 prin intermediul unui bistabil.

Circuitul de generare a semnalului de tact are o importanță deosebită în sistemul Z80, deoarece trebuie respectați parametrii, ca de exemplu timpii de creștere și de scădere ai semnalului de tact. Nerespectarea acestor parametri duce la funcțio-

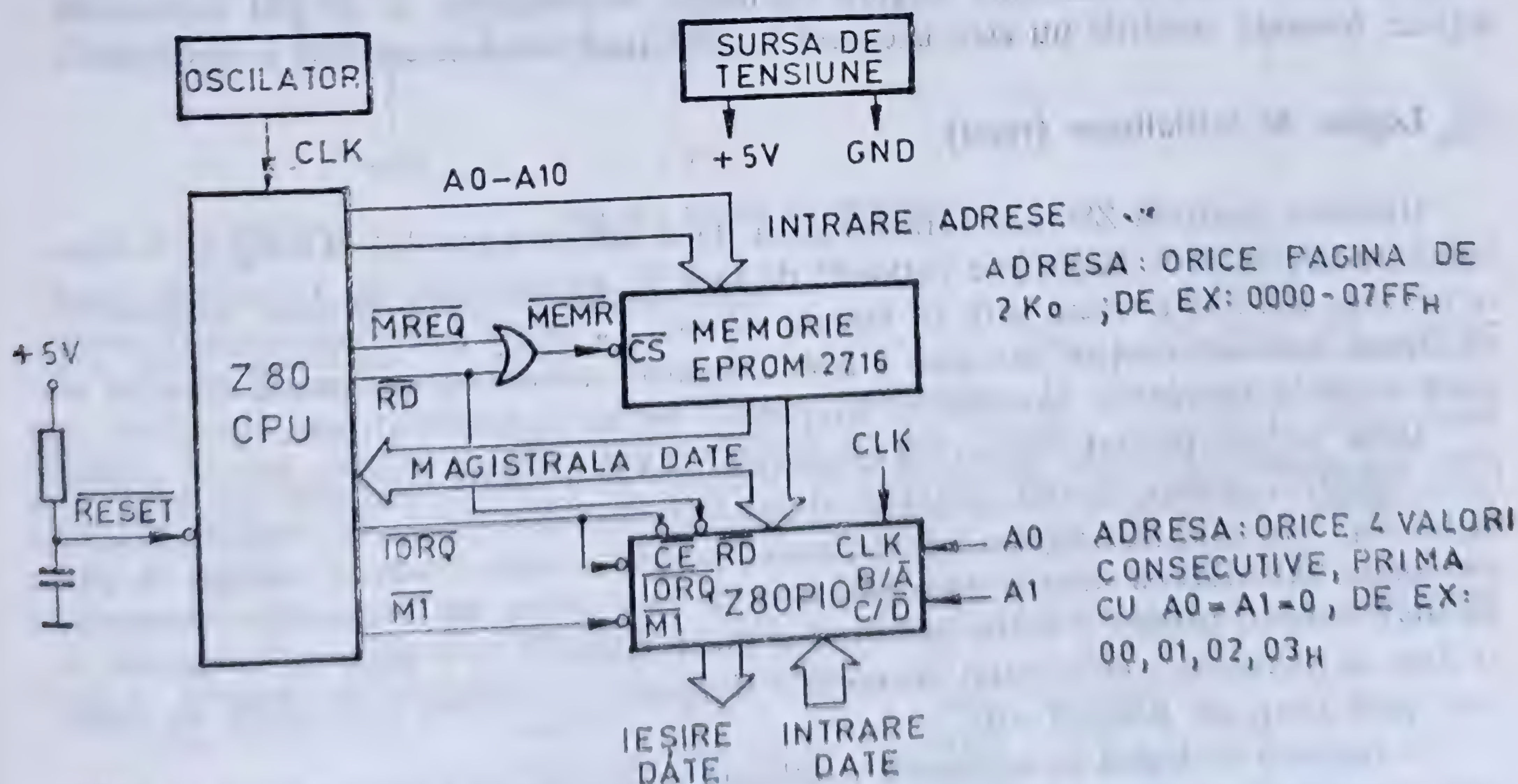


Fig. 8.1 Sistem Z80 minim.



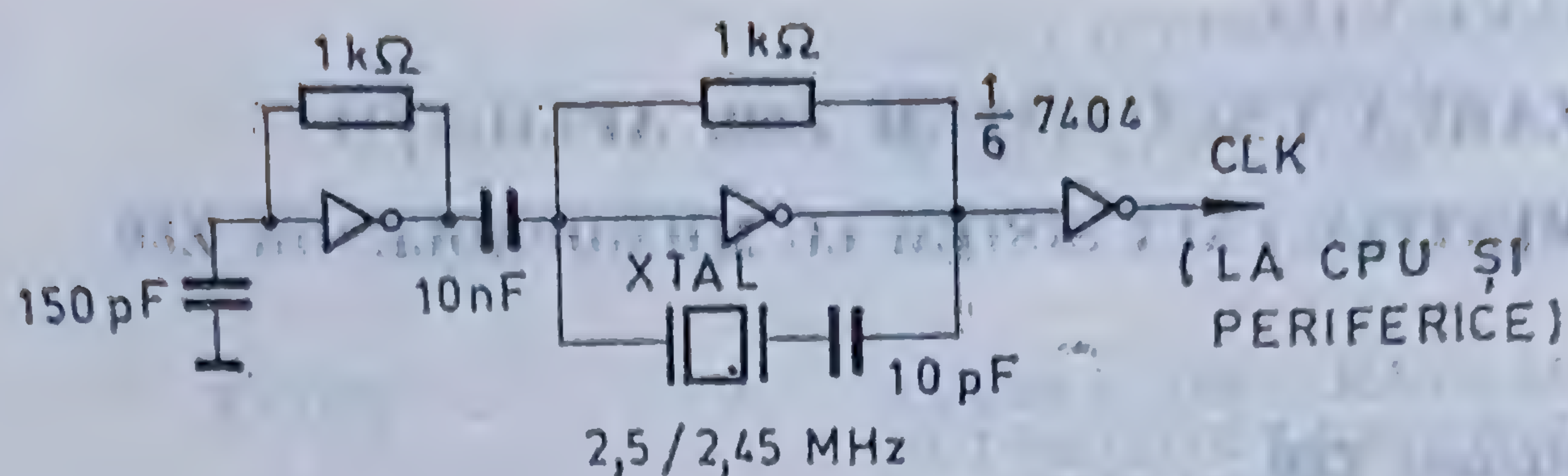


Fig. 8.2 Oscilator cu cristal de cuarț de 2,5 MHz.

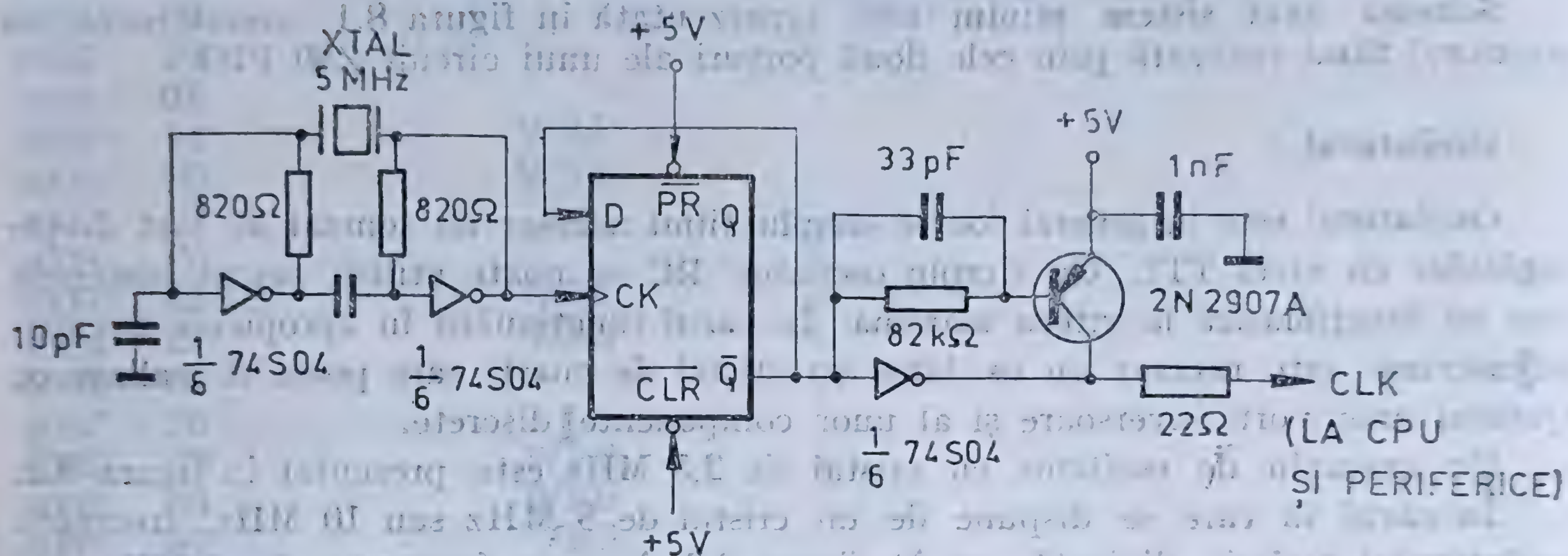


Fig. 8.3 Oscilator cu cristal de cuarț de 5 MHz.

narea incorectă a unității centrale și a circuitelor sistemului. O variantă de generator de tact care asigură timpi de creștere și de scădere a semnalului de tact de 30 ns la o sarcină de 150 pF este prezentată în figura de mai sus. Divizarea cu 2 a frecvenței prin intermediul bistabilului asigură un factor de umplere de 50% al semnalului de tact (această condiție nu este necesară pentru funcționarea corectă a circuitului).

### Logica de inițializare (reset)

Unitatea centrală Z80 are caracteristica de a aduce semnalul  $\overline{MREQ}$  la o stare logică nedeterminată, timp de o perioadă de tact  $T$ , aproximativ la 3 perioade după ce intrarea  $\overline{RESET}$  a trecut la 0, în timpul stărilor  $T_0$  sau  $T_1$  ale unui ciclu de mașină. Dacă sistemul conține memorii RAM dinamice, acest fapt ar putea duce la un scurt acces la memoriile dinamice și conținutul lor ar putea fi alterat.

Dacă trebuie păstrat conținutul memoriilor RAM dinamic după aplicarea semnalului  $\overline{RESET}$ , atunci frontul căzător al lui trebuie sincronizat cu frontul căzător al lui  $\overline{M}_1$ . Circuitul din figura 8.4 realizează această sincronizare și asigură în plus un semnal  $\overline{RESET}$  cu durata de o perioadă de tact, chiar la acționarea butonului  $\overline{RESET}$  extern, pentru a evita menținerea liniei  $\overline{RESET}$  la 0 logic pe o durată ce ar duce la pierderea conținutului memoriilor dinamice, prin lipsa semnalelor de reîmproșare (atât timp cât  $\overline{RESET}=0$ ).

O variantă de logică de inițializare, utilizând monostabilul 74123 în locul lui 74121 este prezentată în figura 8.5.



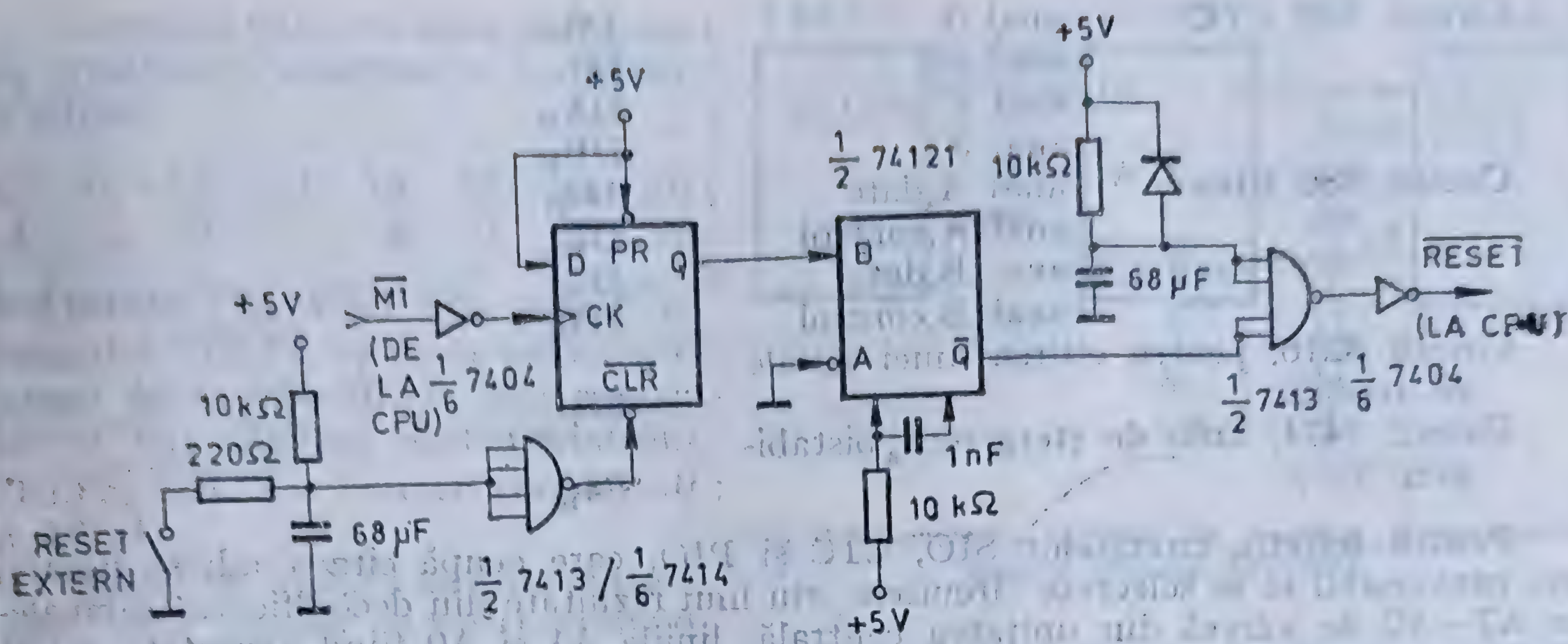


Fig. 8.4 Logică de inițializare la punerea sub tensiune, sau manuală, prin buton.

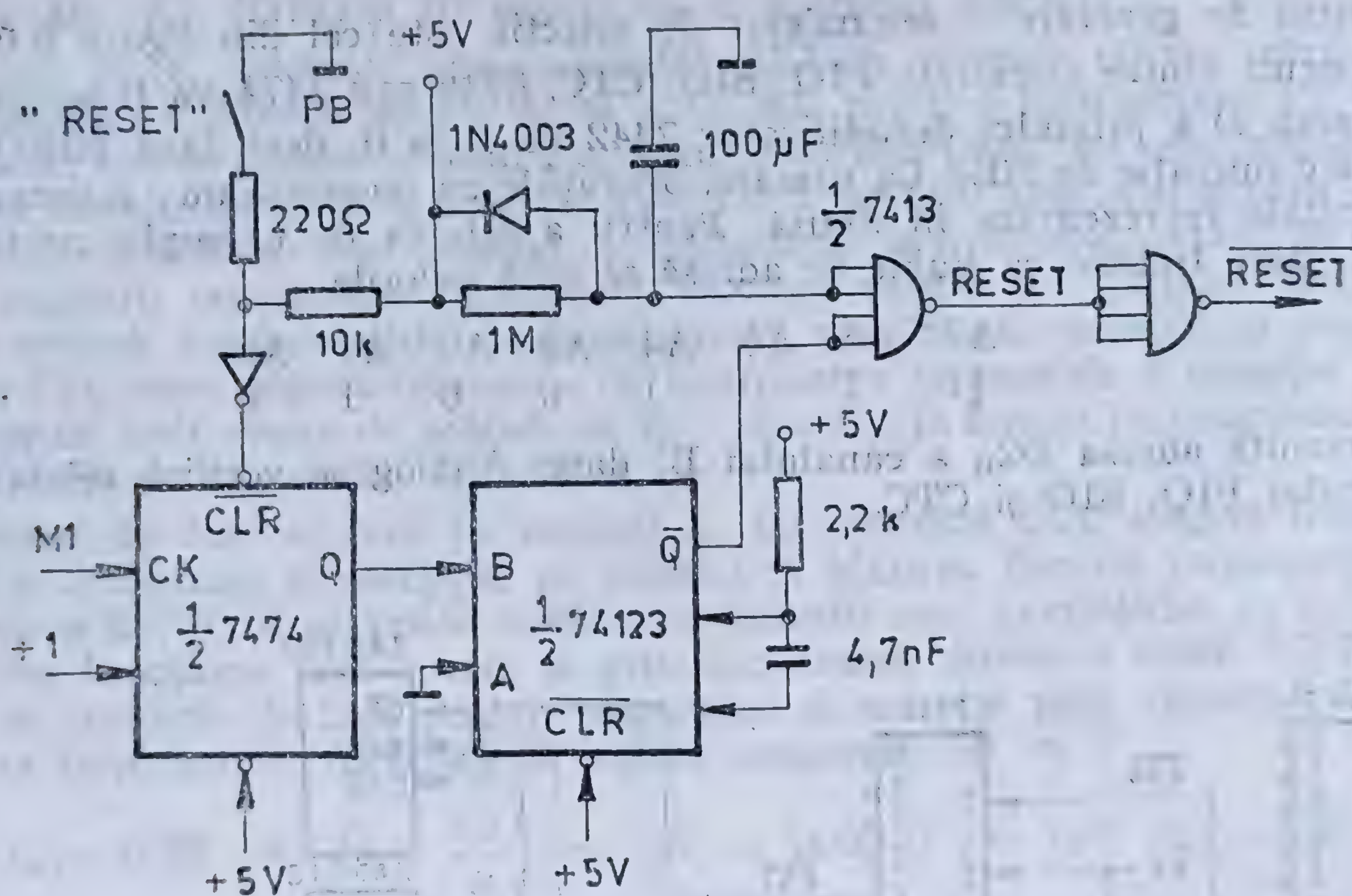


Fig. 8.5 Logică de inițializare.

### Selecția dispozitivelor de intrare/ieșire

Selecția dispozitivelor de intrare/ieșire, ca și a memoriilor, se poate face complet prin utilizarea unor circuite de decodificare a adreselor. În cazul celor mai simple sisteme, se pot elimina parțial sau complet aceste circuite, conectând direct linii de adresă din unitatea centrală la intrările de selecție ale dispozitivelor de I/E sau ale memoriilor, cu condiția să se aleagă pentru fiecare dispozitiv sau memorie o astfel de adresă (zonă de adrese) pentru care să nu se selecteze nici un alt dispozitiv sau memorie. Sînt posibile și variante intermediare între aceste două cazuri extreme.

Selecția dispozitivelor de intrare/ieșire se va exemplifica pentru un sistem în care se impun următoarele adrese:

Circuit Z80 SIO:	canal A, date:	DC <sub>H</sub>
	canal B, date:	DD <sub>H</sub>
	canal A, control:	DE <sub>H</sub>
	canal B, control:	DF <sub>H</sub>



Circuit 7474, linia de ștergere a bistabilului : F4<sub>H</sub>

A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	1	0	0	1	0

The diagram illustrates a Z80 microprocessor system for a console. It features three 7442 BCD-to-7-segment decoders, three Z80 chips (PIO, CTC, SIO), a 7474 flip-flop, an 8216 memory controller, and two LEDs (BC252 and BC171). The PIO is connected to the CTC and SIO. The SIO is connected to the 7474 and the 8216. The 8216 is connected to the Z80. The Z80 is connected to the console output. The console output is connected to the LEDs. The circuit is powered by +5V and -5V.

**Fig. 8.6** Selecția dispozitivelor de I/E.



Semnalul q61 este adus la 0 pentru următoarea combinație a liniilor de adresă:

A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	1	0	1	X	X

decî pentru F4<sub>H</sub>, F5<sub>H</sub>, F6<sub>H</sub> sau F7<sub>H</sub>. Semnalul SELF4 va avea valoarea 0 numai dacă A1=A0=0, deci pentru adresa F4<sub>H</sub>. Analog, pentru semnalul SELF5, SELF5=0 numai cînd A1=0 și A0=1.

Semnalul SELF4 servește la ștergerea unui bistabil de tip D, printr-o instrucțiune IN A, (F4H) sau OUT (F4H), A, fără ca octetul citit sau înscris să aibă vreo semnificație. Rolul bistabilului va rezulta în discuția circuitului de selecție a memoriei. Liniile A0—A7, IORQ și D7 provin de la unitatea centrală.

Circuitul 8216, selectat prin semnalul SELF5, permite citirea stării logice a liniei de recepție a datelor seriale de la SIO, RI, printr-o instrucțiune IN A, (F5H). Starea acestei linii se citește în bitul 7 al acumulatorului (linia RI este conectată prin intermediul circuitului 8216 la linia de date D7 a unității centrale) pentru a măsura durata bitului de start al unui caracter recepționat de la o consolă serială, la începutul stabilirii legăturii cu consola. Măsurarea duratei acestui bit permite determinarea vitezei de schimb a informațiilor pe linia serială, stabilită de consolă, și programarea circuitului CTC care asigură frecvența de tact pentru transmisie și recepție la circuitul SIO, astfel încît viteza de schimb să fie aceeași și în sistem (v. programul prezentat la aplicații ale circuitului Z80 CTC).

Schimbul de date se face pe canalul A, iar circuitul CTC asigură frecvența de tact pentru transmisie și recepție, pe canalul 0. Mărirea duratei impulsurilor furnizate de ieșirea ZC/TO 0 se poate realiza cu ajutorul unui monostabil, ca în figura 8.7 (circuitul va funcționa însă corect și prin conectarea directă a ieșirii ZC/TO 0 a lui Z80 CTC la intrările de tact pentru transmisie și recepție prin circuitul Z80 SIO).

Durata impulsurilor furnizate la ieșirea monostabilului va fi:

$$t_w = 0,32 \cdot RC \left( 1 + \frac{0,7}{R} \right), \text{ cu } [R] = 1k\Omega, [C] = 1pF [t_w] = 1ns$$

Adaptarea nivelului 1 logic de pe linia de recepție sau transmisie a consolei (-3 ÷ -12V) și a celui de 0 logic (+3 ÷ +12V), cu nivelul 0 sau 1 logic TTL, se poate efectua fie prin circuitele integrate specializate MC 1408, MC 1409, fie prin intermediul unor tranzistoare, ca în figura 8.6 de mai sus.

### Selecția liniară

În cazul în care sistemul conține un număr restrîns de dispozitive de I/E, se poate face selecția lor fără circuite de decodificare, conectînd direct linii de adresă ale unității centrale la intrările de selecție ale circuitelor. Un exemplu este prezentat în figura 8.8, în care circuitul Z80 PIO este selectat ori de cîte ori A7=0, indiferent de starea logică a celorlalte linii de adresă. Analog, circuitul Z80 CTC va fi selectat cînd A6=0. Ca urmare se poate alege o adresă pentru fiecare dintre aceste două circuite, dînd valori arbitrare biților care corespund liniilor A5 ÷ A2. Liniile A1 și A0 selectează una dintre cele 4 locații ale fiecărui circuit, în care se înscrie un cuvînt sau din care se citește un cuvînt de 8 biți la executarea unei instrucțiuni de ieșire sau de intrare cu A7=0 sau A6=0.

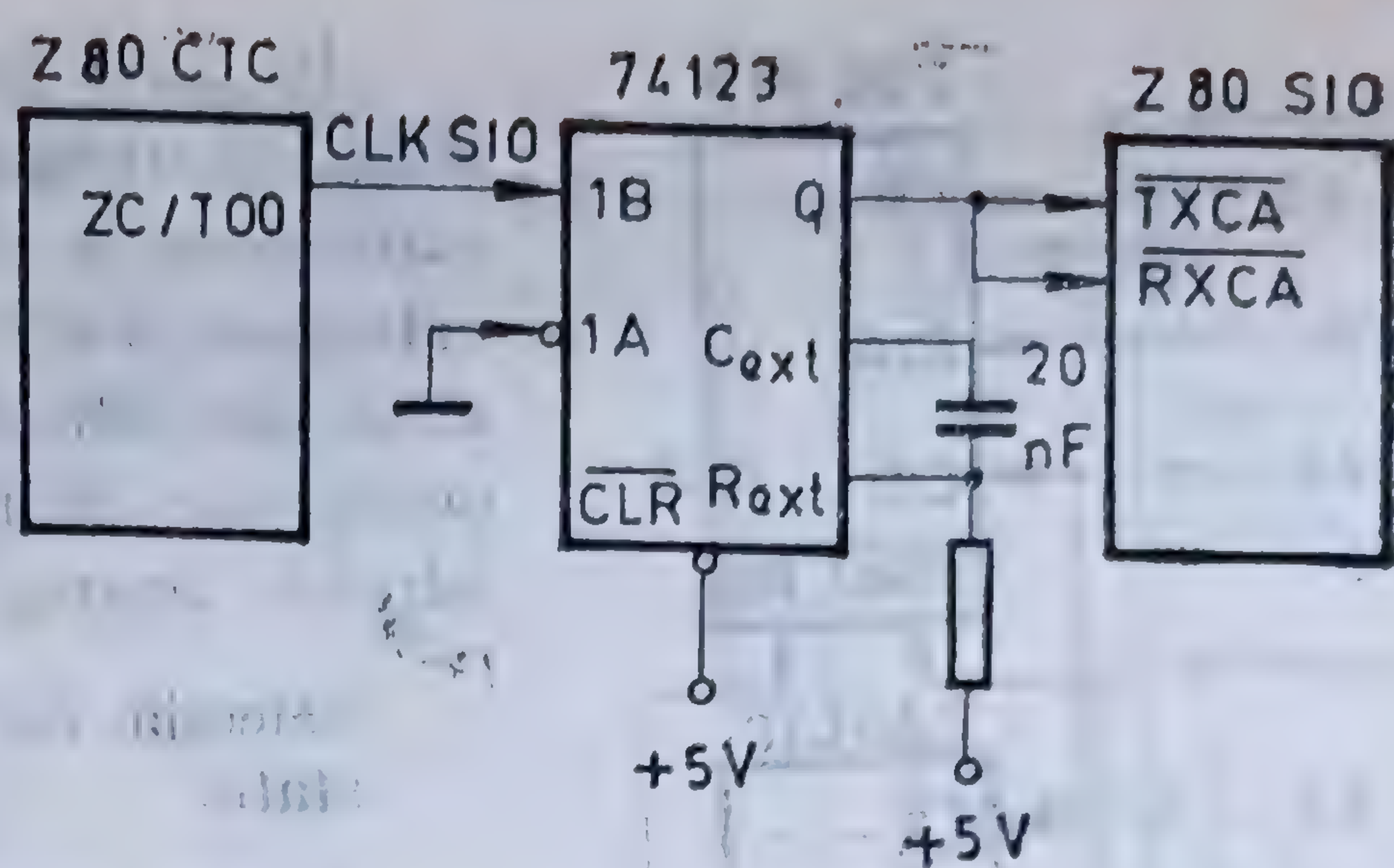


Fig. 8.7 Generarea tactului de recepție și transmisie pentru circuitul Z80 SIO.







Dacă bistabilul D, al cărui rol va fi explicat în continuare, este șters, deci  $Q=0$ , atunci selecția unui circuit de memorie se face doar cu ajutorul semnalelor  $A_{15}-A_{11}$  și  $\overline{MREQ}$ . Vom avea de exemplu  $Q_{24}=0$  (ieșirea 0 a celui de al doilea decodificator) dacă adresele au configurația

$A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9$   
1 1 0 0 0 X X

$A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$   
X X X X X X X X X

deci pentru orice adresă între  $C000_H$  și  $C7FF_H$ .

Analog se determină adresele pentru care  $Q_{25}-Q_{28}$  iau valoarea 0, selectînd astfel circuitele EPROM.

Semnalul  $Q_{31}$  este 0 pentru orice adresă în domeniul  $F800-FFFF_H$ , deci pentru o zonă de 2 ko.

Ca urmare, dacă se conectează la linia de selecție a unei memorii de 1 ko, adresată cu 10 linii, de exemplu  $A_0-A_9$ , linia  $A_{10}$  nu va fi utilizată iar valoarea ei va fi indiferentă. De exemplu adresele  $F800$  (cu  $A_{10}=0$ ) și  $FC00$  (cu  $A_{10}=1$ ) vor indica, ambele, prima locație de memorie a circuitului de 1 ko RAM static. Dacă se dorește utilizarea completă a spațiului de 2 ko pentru care  $Q_{31}=0$ , se poate completa decodificarea cu ajutorul unor porți ca în figura 8.10.

O variantă de generare a semnalelor de selecție este de a obține semnale de selecție pentru spații de memorie de 1 ko și de a le combina câte 2 dacă memoria adresată este de 2 ko, ca în figura 8.11.

Proiectarea circuitelor de decodificare se face pornind de la adresele dorite pentru memorii, și efectuînd conexiunile la ieșirile decodificatoarelor care sînt active pentru adresele impuse.

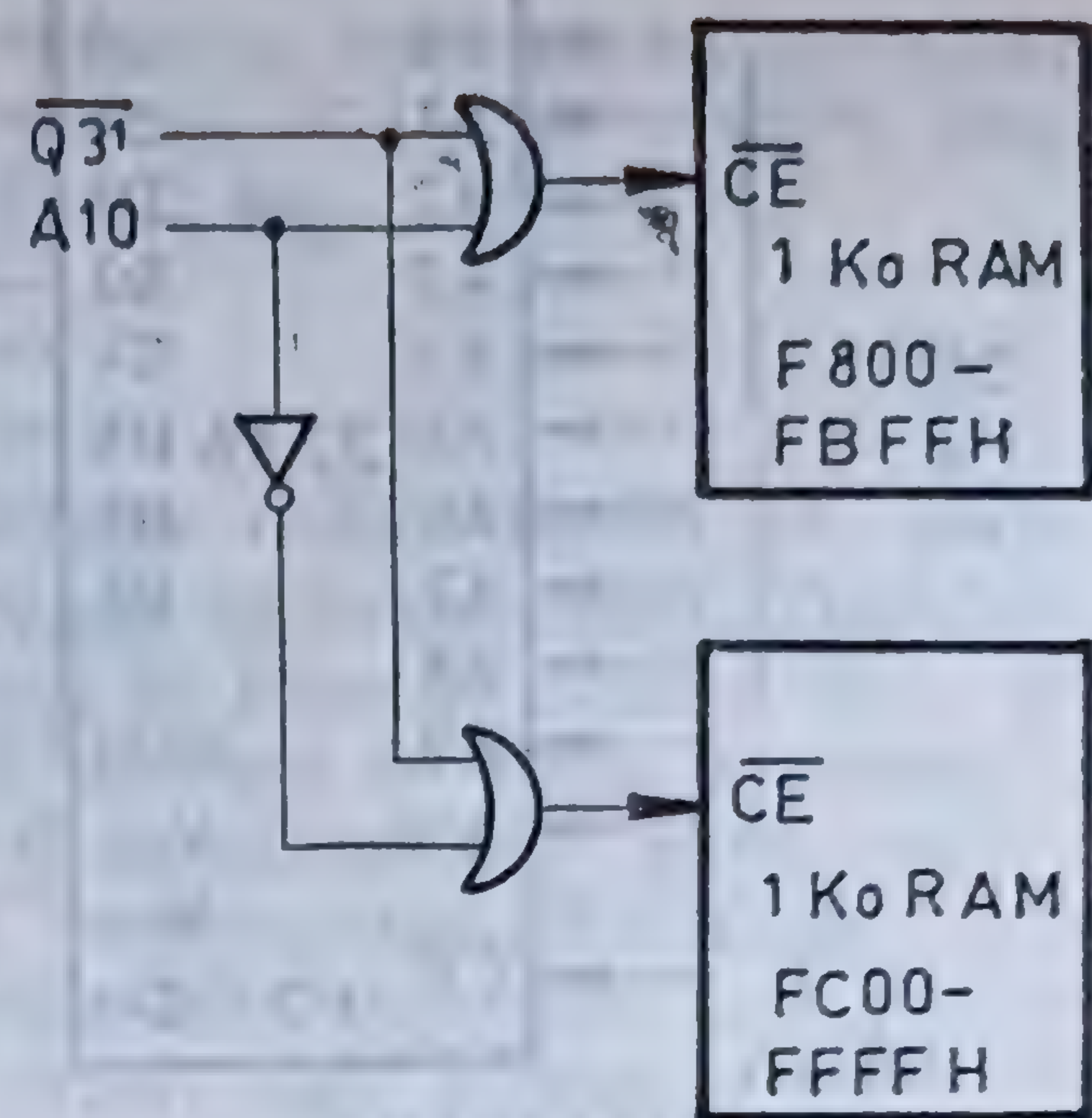


Fig. 8.10 Complectarea decodificării cu porți.

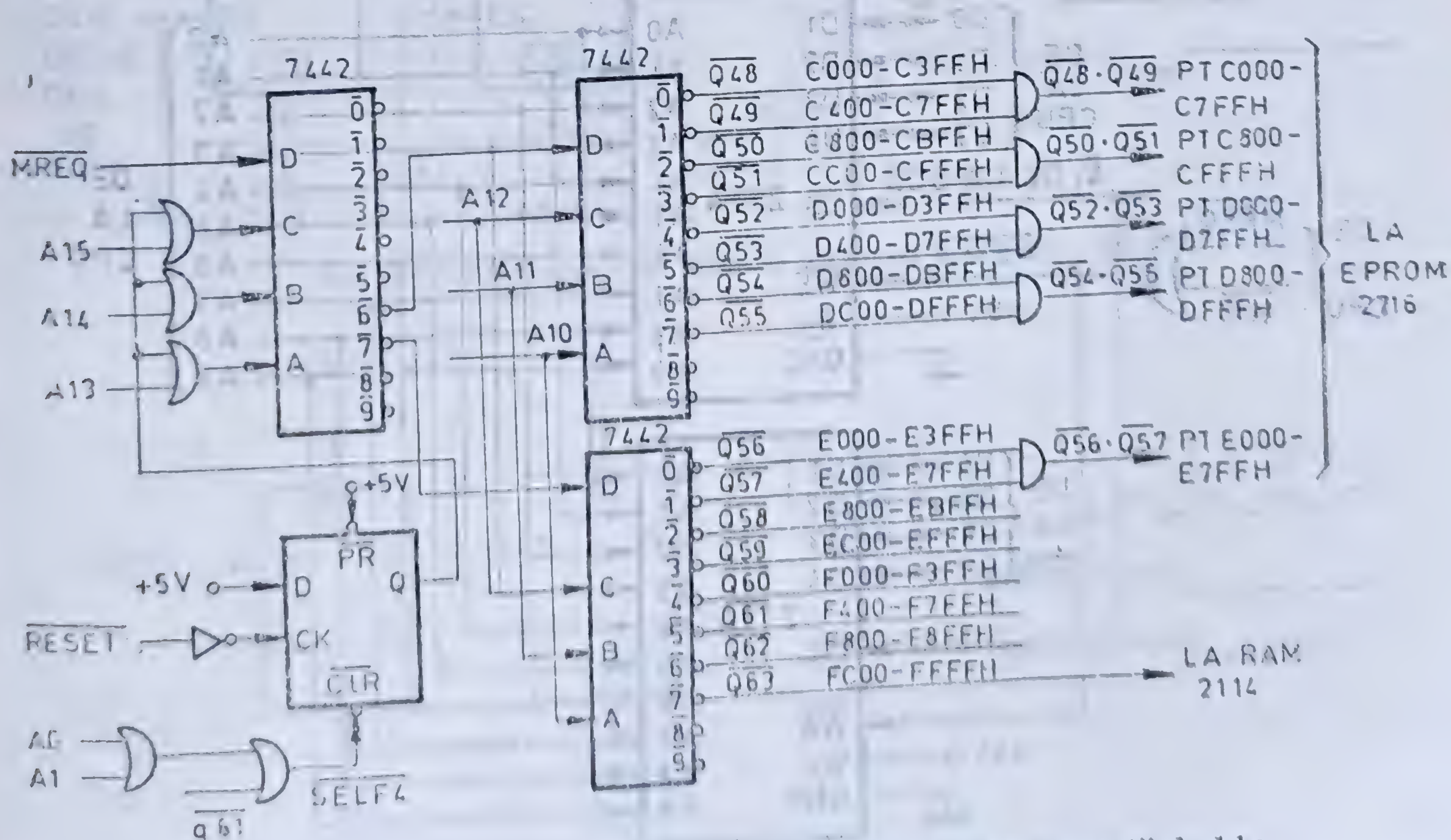


Fig. 8.11 Generarea semnalelor de selecție a memoriilor, pentru spații de 1 ko.



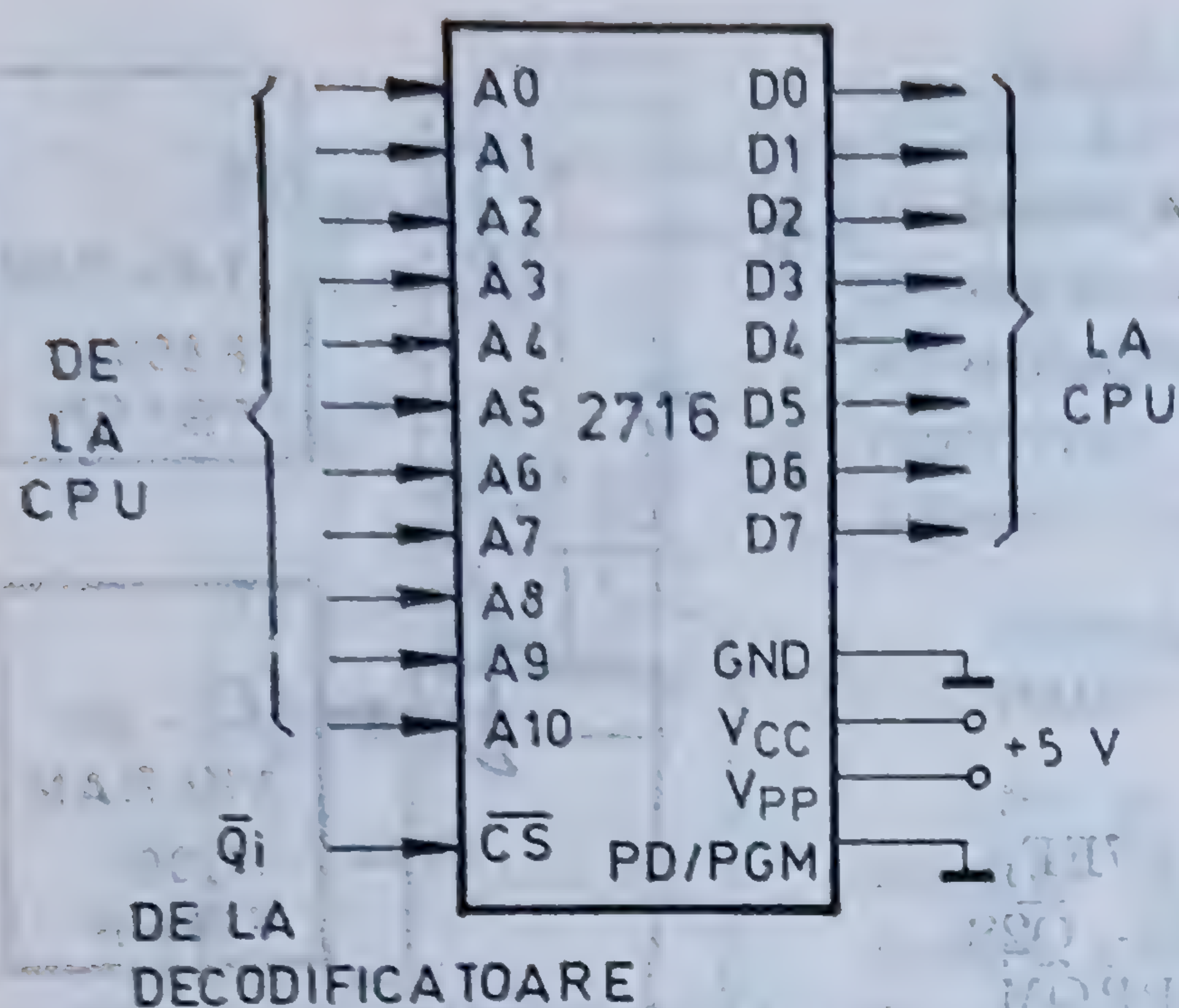


Fig. 8.12 Conectarea memoriilor EPROM 2716.

ADRESA	CODUL	ETICHETA	COD	OPERATIE	COMENTARIU
E000	C303E0	JMP RESET			SALT LA E003H, PC VA CONTINE E003H
E003	DBF4	RESET:	IN A, PSYS		FACE SELF4=0, DECI Q=0

După saltul la E003<sub>H</sub>, forțarea adresei din care se decodifică semnalele de selecție ale memoriilor, nu mai este necesară, și bistabilul este șters prin apariția adresei F4<sub>H</sub> pe liniile A7—A0, la execuția instrucțiunii IN A, (F4H) (adresa F4<sub>H</sub>=PSYS corespunde locației denumite "PORT DE SISTEM"). Inscrierea bistabilului se poate realiza numai la o nouă inițializare, prin RESET=0.

Semnalele de selecție generate cu ajutorul decodificatoarelor se aplică direct pe liniile de selecție ale circuitelor de memorie, ca în figura 8.12 pentru memoriile EPROM 2716 și în figura 8.13 pentru memoriile RAM 2114.

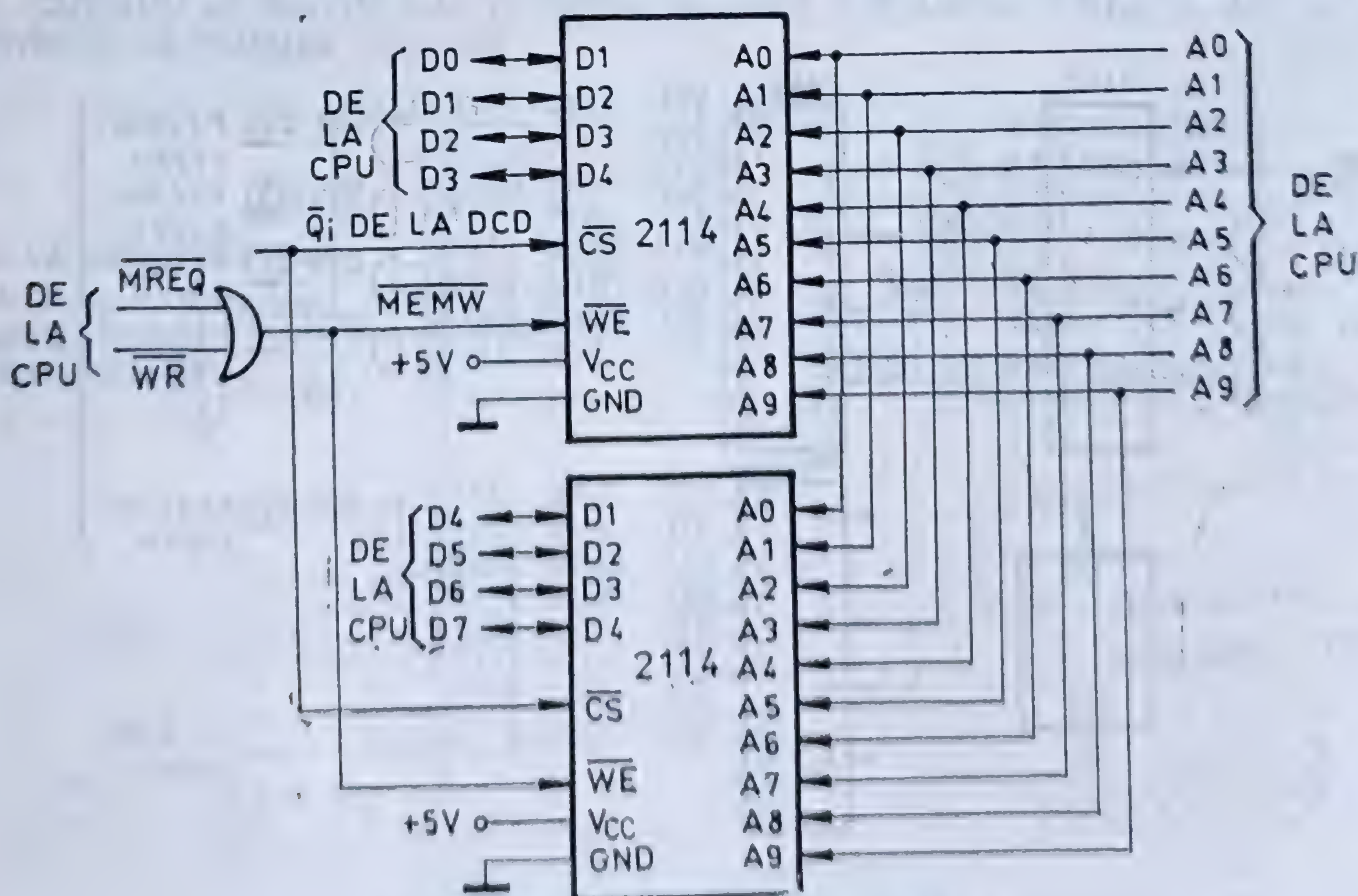


Fig. 8.13 Conectarea memoriilor RAM 2114.



Numărul de linii de adresă conectate direct de la unitatea centrală la memorie, depinde de capacitatea acesteia: 10 linii, A0—A9, pentru 2114 ( $2^{10}=1024$ , deci 1 ko), și 11 linii, A0—A10, pentru 2716 ( $2^{11}=2048$ , deci 2 ko).

### Controlul timpului de acces la memorie

Pentru unele aplicații, utilizarea unor memorii mai lente reduce costul lor. Linia  $\overline{\text{WAIT}}$  a unității centrale permite funcționarea cu memorii de orice viteză. Cerințele relative la timpul de acces la memorie trebuie respectate în timpul ciclului M1, de aducere din memorie a codului operației. Toate celelalte operații de acces la memorie au o jumătate de perioadă de tact în plus, pentru a fi terminat. Din acest motiv, în unele situații poate fi necesară adăugarea unei stări de așteptare (WAIT) la ciclul M1, ceea ce permite utilizarea memoriilor lente. Figura 8.14 reprezintă un circuit care realizează această funcție, iar figura 8.15, un circuit care adaugă o stare de așteptare la orice operație de acces la memorie.

### Amplificarea/separarea semnalelor emise sau recepționate de unitatea centrală

În cazul sistemelor cu mai multe circuite de memorie sau de I/E, poate fi necesară amplificarea unor semnale din sistem. De cele mai multe ori, semnalele  $\overline{\text{M1}}$ ,  $\overline{\text{MREQ}}$ ,  $\overline{\text{IORQ}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$  și  $\overline{\text{RFSH}}$  sînt amplificate cu ajutorul unor porți inversoare, sau al unor circuite amplificatoare cu 3 stări (de exemplu 74125), ca în figura 8.16.

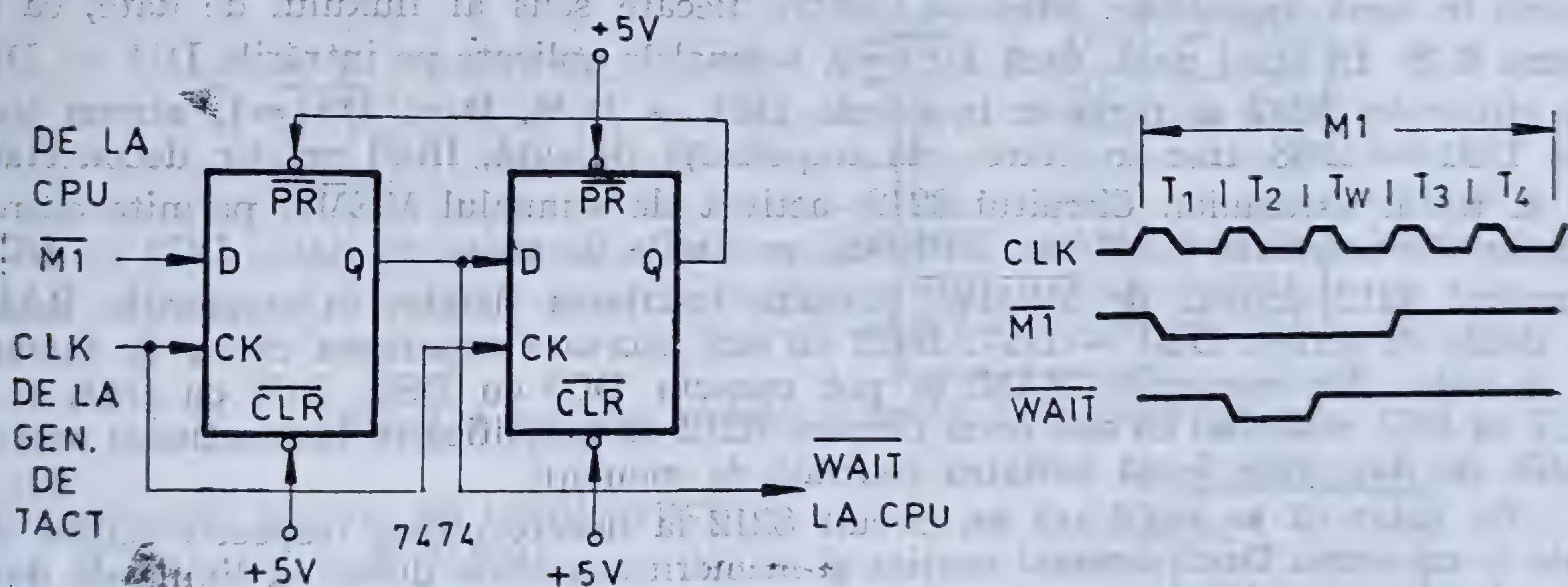


Fig. 8.14 Adăugarea unei stări de așteptare la un ciclu M1.

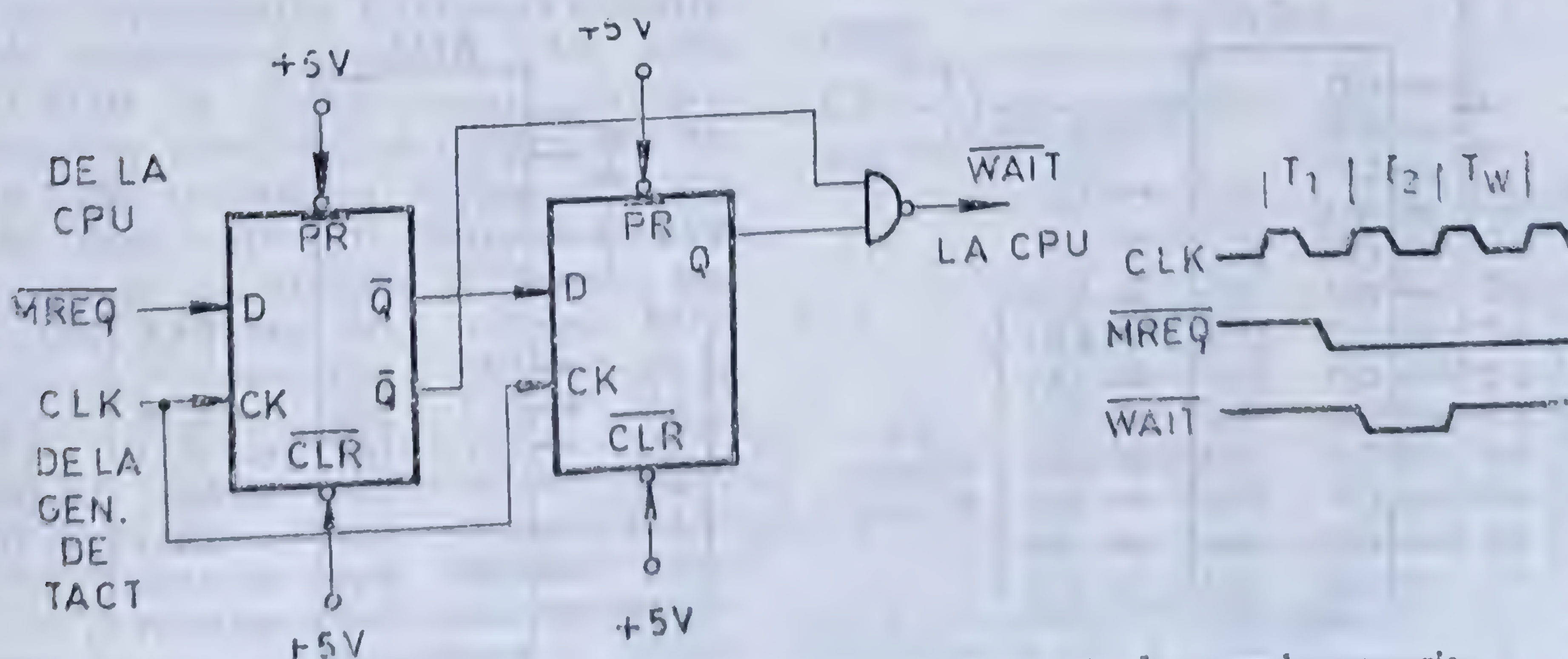


Fig. 8.15 Adăugarea unei stări de așteptare la fiecare ciclu de acces la memorie.



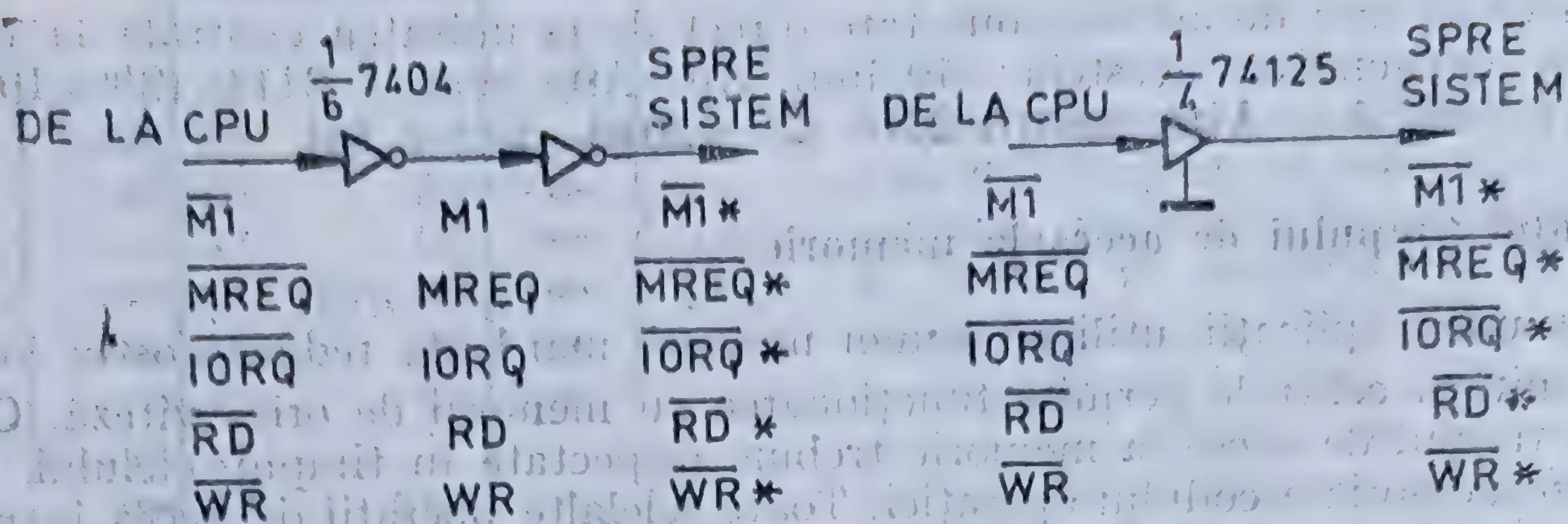


Fig. 8.16 Amplificarea/separarea unor semnale de control emise de unitatea centrală.

Separarea semnalelor emise sau recepționate de unitatea centrală asigură și protecție împotriva avariilor care pot să apară în sistem sau împotriva perturbațiilor care apar dacă liniile de legătură între circuite au o lungime prea mare.

Amplificarea semnalelor de adresă emise de unitatea centrală se realizează în mod obișnuit cu circuite 74125, 8216/8226 sau 8212. Varianta cu circuite 8212, într-o conexiune în care sînt transparente la nivelele logice aplicate pe intrări, este prezentată în figura 8.17.

Liniile de date pot fi de asemenea separate de restul sistemului, uneori din necesitatea de a separa calea de intrare a datelor în memoriile RAM de calea de ieșire a lor. În acest caz, magistrala de date bidirecțională a unității centrale se separă în două magistrale, cîte una pentru fiecare sens al fluxului de date, ca în figura 8.18. În acest mod, dacă  $\overline{DS1}=0$ , semnalele aplicate pe intrările DI1 — DI8 ale circuitelor 8212 se regăsesc la ieșirile D01 — D08. Dacă  $\overline{DS1}=1$ , atunci ieșirile D01 — D08 trec în starea de impedanță ridicată, fiind practic deconectate de la restul sistemului. Circuitul 8212, activat de semnalul  $\overline{MEMR}$  permite citirea datelor din memoria RAM sau EPROM, pe liniile de citire de date, DC0 — DC7. Circuitul 8212 activat de  $\overline{MEMW}$ , permite înscrierea datelor în memoriile RAM, pe liniile de scriere DS0 — DS7. Dacă nu este necesară separarea căilor de intrare și de ieșire din memoriile RAM, se pot conecta DC0 cu DS0, DC1 cu DS1, ..., DC7 cu DS7, realizînd cu cele două circuite 8212 un amplificator bidirecțional pentru liniile de date care leagă unitatea centrală de memorii.

De notat că se validează un circuit 8212 la fiecare citire, respectiv scriere de date în memorie. Dacă sistemul conține și memorii conectate direct la liniile de date ale microprocesorului, sau pe alte căi, sistemul nu va funcționa corect cînd se citesc

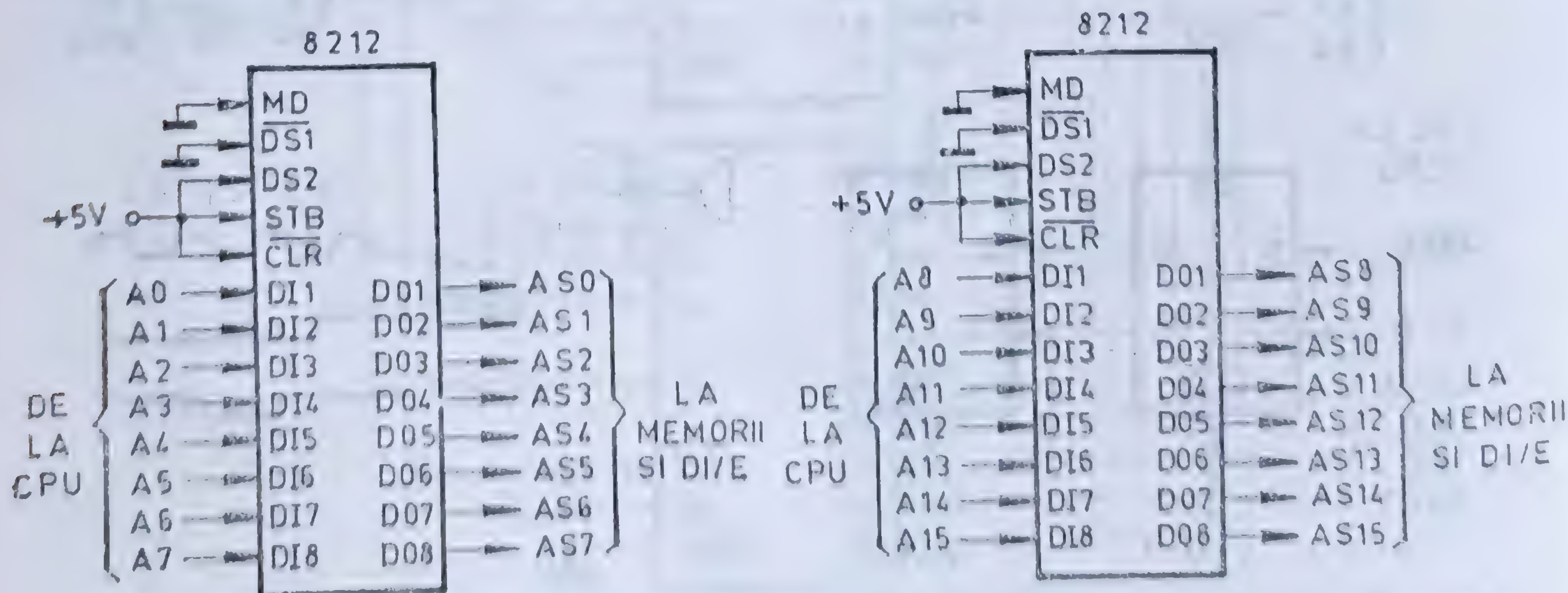


Fig. 8.17 Amplificarea/separarea de sistem a semnalelor de adresă emise de unitatea centrală.



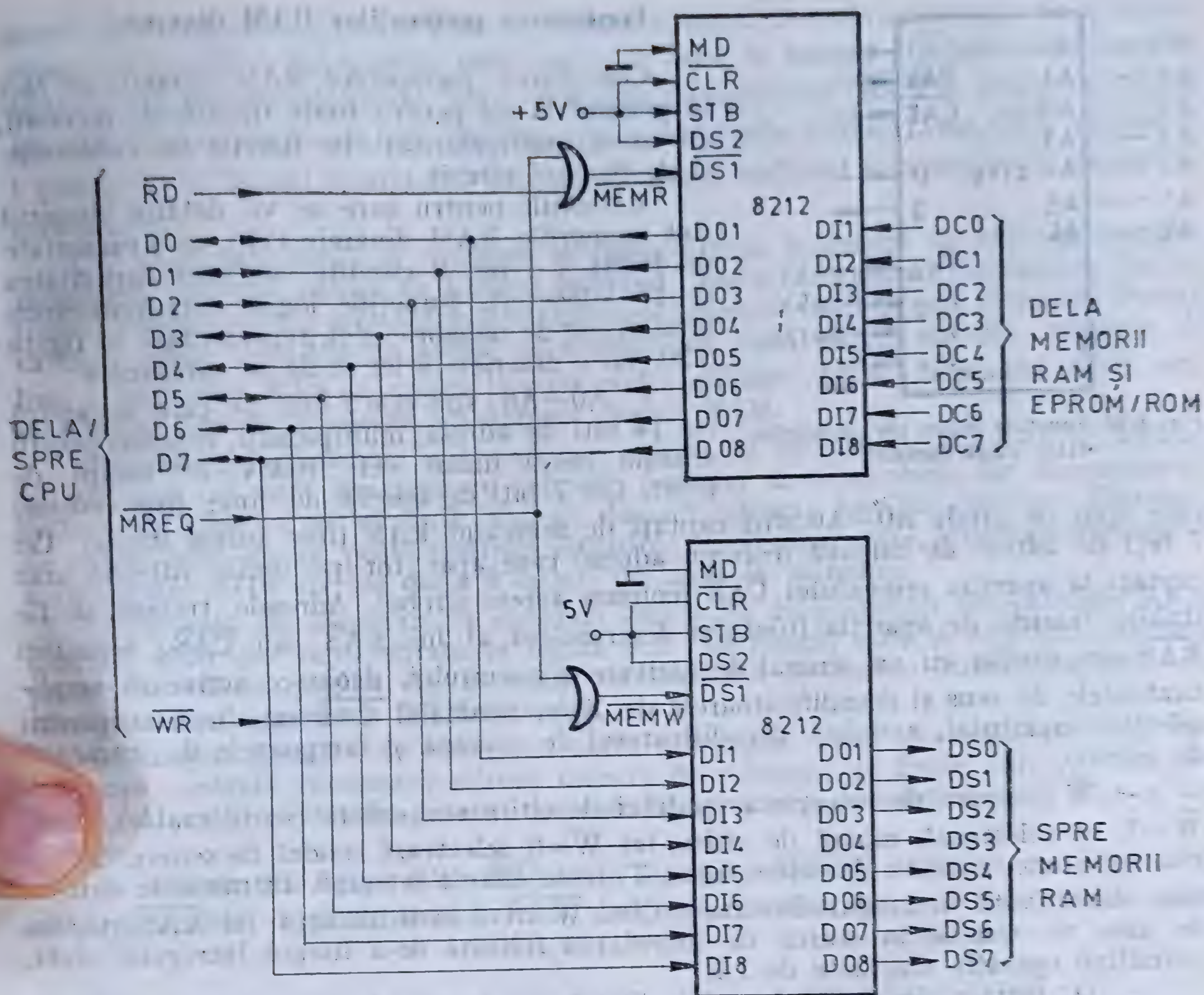


Fig. 8.18 Separarea intrărilor și ieșirilor de la memorii.

acele memorii, pentru că ieșirile circuitului 8212 activat de  $\overline{\text{MEMR}}$  ies din starea de impedanță ridicată. De exemplu, dacă sistemul conține memorii RAM dinamic în jumătatea inferioară a spațiului de memorie și memorii RAM static și EPROM în jumătatea superioară, acestea din urmă utilizând circuitele din figura 8.18, trebuie ca ieșirea din starea de impedanță ridicată a circuitului 8212 activat de  $\overline{\text{MEMR}}$  să se facă numai când  $A_{15}=1$ , ca în figura 8.19. Deschiderea circuitului 8212 activat de  $\overline{\text{MEMW}}$  nu deranjează, pentru că memoriile pe intrările cărora se aplică datele care trec prin circuitul 8212, nu sînt selectate, în cazul înscririi unor date în jumătatea inferioară a memoriei. Aceleași funcții pot fi realizate și cu alte circuite (de exemplu cu 8216 sau 74125).

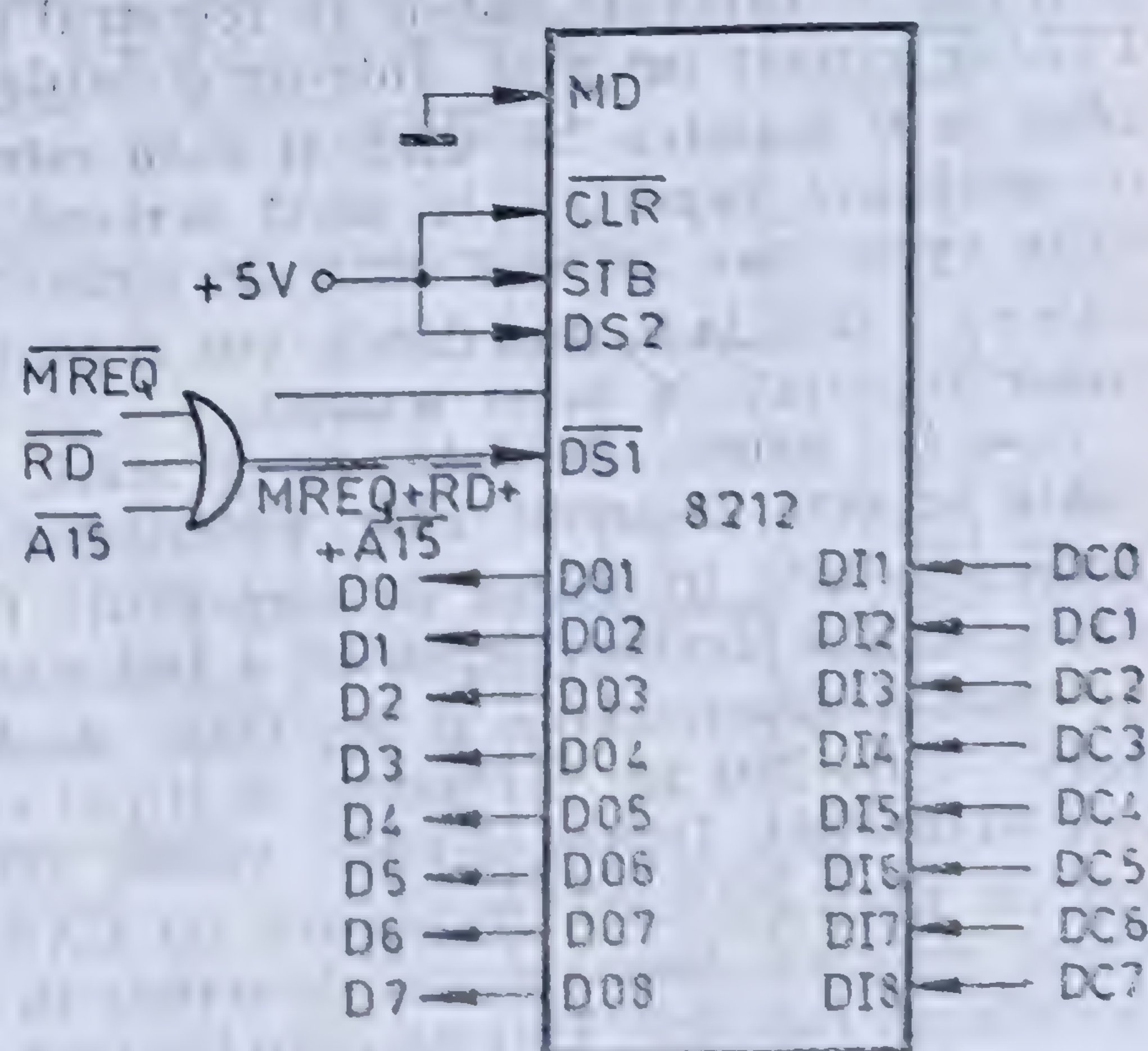


Fig. 8.19 Condiționarea deschiderii căii de citire a datelor dintr-o zonă a memoriei (se deschide pentru jumătatea superioară).



### Conectarea memoriilor RAM dinamic

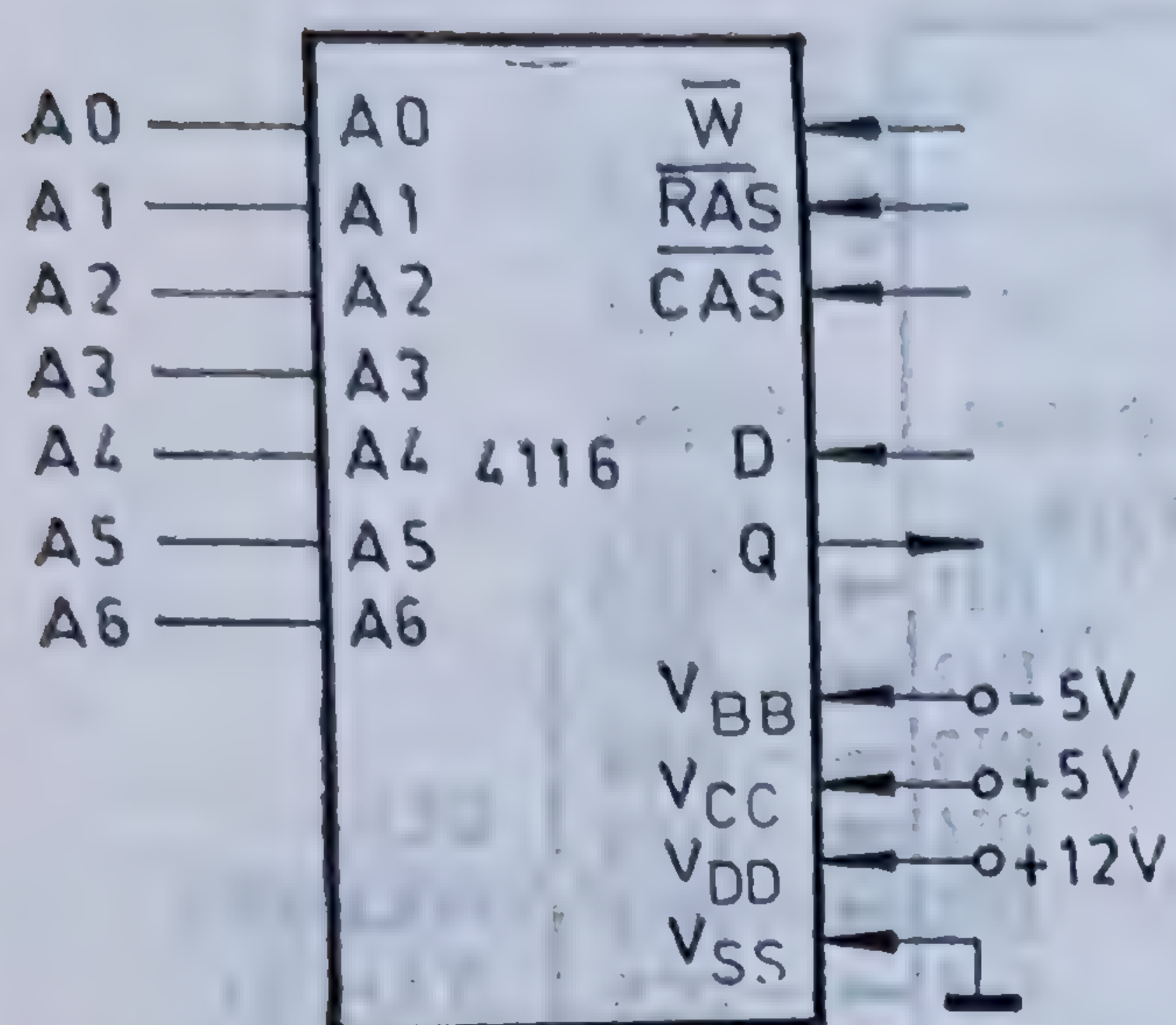


Fig. 8.20 Funcțiile logice ale circuitului 4116, RAM dinamic.

Conectarea memoriilor RAM dinamic se face în general la fel pentru toate tipurile de memorii, cu unele particularități în funcție de caracteristicile fiecărui circuit.

Memoriile pentru care se va detalia circuitul sînt memoriile RAM dinamic 4116, cu o capacitate de  $16384 \times 1$  bit. 8 circuite vor avea capacitatea de 16 kiloocteți. Funcțiile logice corespunzătoare unui circuit de memorie sînt reprezentate în figura 8.20, iar o descriere a lor se dă în continuare:

— A0—A6: sînt cele 7 linii pe care se aplică cei 14 biți de adresă, multiplexați, necesari pentru selecția uneia dintre cele  $16384=2^{14}$  locații de 1 bit. Cei 7 biți de adresă de linie (row address) care apar pe liniile A0—A6 sînt captați de semnalul  $\overline{\text{RAS}}$  (row address strobe). Cei 7 biți de adresă de coloană (column address) care apar tot pe liniile A0—A6 sînt captați la apariția semnalului  $\overline{\text{CAS}}$  (column address strobe). Adresele trebuie să fie stabile înainte de apariția frontului descrescător al lui  $\overline{\text{RAS}}$  sau  $\overline{\text{CAS}}$ . Semnalul  $\overline{\text{RAS}}$  este similar cu un semnal de activare a circuitului, deoarece activează amplificatoarele de sens și decodificatoarele de linie. Semnalul  $\overline{\text{CAS}}$  este utilizat pentru selecția circuitului, activînd decodicatorul de coloană și tamponanele de intrare și de ieșire.

—  $\overline{\text{W}}$ : intrare de selecție a modului de citire sau scriere (write enable). Dacă  $\overline{\text{W}}=1$ , se selectează modul de citire, iar  $\overline{\text{W}}=0$  selectează modul de scriere. Intrarea se poate conecta la orice ieșire TTL de circuit integrat. Intrarea de date D este dezactivată în modul de citire. Cînd  $\overline{\text{W}}$  trece la 0 înaintea lui  $\overline{\text{CAS}}$ , ieșirea de date va rămîne în starea de impedanță ridicată de-a lungul întregului ciclu, permițînd operații obișnuite de I/E.

— D: intrare de date (data in). Datele se înscriu în timpul unui ciclu de scriere sau de citire-modificare-scriere. Ultimul dintre fronturile căzătoare ale lui  $\overline{\text{CAS}}$  sau  $\overline{\text{W}}$  captează datele în registrul înglobat, la care se pot conecta direct ieșiri TTL de circuit integrat. Într-un ciclu de scriere în avans (early write cycle),  $\overline{\text{W}}$  este adus la 0 înaintea lui  $\overline{\text{CAS}}$  și data este captată de  $\overline{\text{CAS}}$ , cu timpi de prestabilire și menținere raportați la acest semnal. Într-un ciclu de scriere întîrziat (delayed write cycle) sau într-un ciclu de citire-modificare-scriere (read-modify-write cycle),  $\overline{\text{CAS}}$  va fi deja la 0, deci datele vor fi captate de  $\overline{\text{W}}$ , cu timpi de prestabilire și menținere raportați la acest semnal.

— Q: ieșirea de date, cu 3 stări, a tamponului de ieșire (data out). Ieșirea poate suporta 2 sarcini TTL. Polaritatea datelor este identică cu cea de la intrare. Ieșirea se află în starea de impedanță ridicată cît timp  $\overline{\text{CAS}}=1$ . Într-un ciclu de citire, ieșirea devine activă cu o întîrziere  $t_{a(c)}$  (interval de timp de validare) față de frontul descrescător al lui  $\overline{\text{CAS}}$ , dacă  $t_{a(R)}$  este respectat (timp de acces de la RAS: 150—250 ns, în funcție de tipul circuitului, care poate fi 4116—15, 4116—20 sau 4116—25). Ieșirea devine validă după ce s-a seurs timpul de acces și rămîne așa cît timp  $\overline{\text{CAS}}=0$ . Revenirea lui  $\overline{\text{CAS}}$  la 1 o face să rămînă în starea de impedanță ridicată. Într-un ciclu de scriere în avans, ieșirea este întotdeauna în starea de impedanță ridicată. Într-un ciclu întîrziat de scriere sau într-un ciclu de citire-modificare-scriere ieșirea va urma secvența dintr-un ciclu de citire.

Refreșarea memoriilor dinamice este o operație necesară pentru a menține datele și trebuie efectuată cel puțin o dată la 2 ms. Deoarece tamponul de ieșire este în



starea de impedanță ridicată pînă la aplicarea lui  $\overline{\text{CAS}}$  activ, secvența de refreșare numai cu semnalul  $\overline{\text{RAS}}$  evită orice ieșire de date în timpul ei. Activarea fiecăreia dintre cele 128 de adrese de linie, realizată cu semnalul  $\overline{\text{RAS}}$  și liniile de adresă A0—A6, determină refreșarea tuturor biților din toate liniile. Linia  $\overline{\text{CAS}}$  rămîne la 1 (inactivă) în timpul acestei secvențe de refreșare, realizînd și o economie de putere consumată.

Un acces mai rapid la memorii se poate obține în modul de lucru pe pagină, menținînd aceeași adresă de linie și activînd adrese de coloană succesive. Astfel, timpul necesar pentru prestabilirea și activarea adreselor de linie este eliminat. Pentru extensie peste cele 128 de adrese de coloană ale unui singur circuit, adresele de linie și semnalul  $\overline{\text{RAS}}$  se aplică la mai multe circuite RAM. Semnalul  $\overline{\text{CAS}}$  este decodificat pentru a selecta un anumit circuit RAM.

Conectarea la tensiune a circuitului trebuie să se facă cu  $V_{\text{BB}}$  (—5 V) aplicat înainte, sau cel mult simultan cu celelalte tensiuni de alimentare, iar deconectarea trebuie să aibă loc astfel încît  $V_{\text{BB}}$  să dispară ultima. În caz contrar, disiparea de putere care are loc poate distruge circuitul. După alimentare, funcționarea corectă a circuitului 4116 are loc după 8 cicluri de acces la memorie.

O variantă de conectare la sistemul Z80 a 32 de kiloocteți de memorie RAM dinamic 4116 este reprezentată în figura 8.21. În figură, A14 și A15 sînt utilizate pentru a selecta pagina de memorie reprezentată de fiecare dintre grupurile de 16 kiloocteți de memorie RAM dinamic.

În timpul operației de refreșare, toate memoriile RAM dinamic vor fi citite. Unitatea centrală furnizează adresa corectă de refreșare pe liniile A0—A6. În sisteme mai extinse, sînt în general necesare tamponare pe magistralele de date și de adrese.

În schema din figura 8.21,  $\overline{\text{CAS}}$  este semnalul  $\overline{\text{MREQ}}$ , întîrziat cu  $2\Delta t$ , încît să devină activ după semnalul  $\overline{\text{RAS}}$ , care este utilizat pentru fixarea adreselor

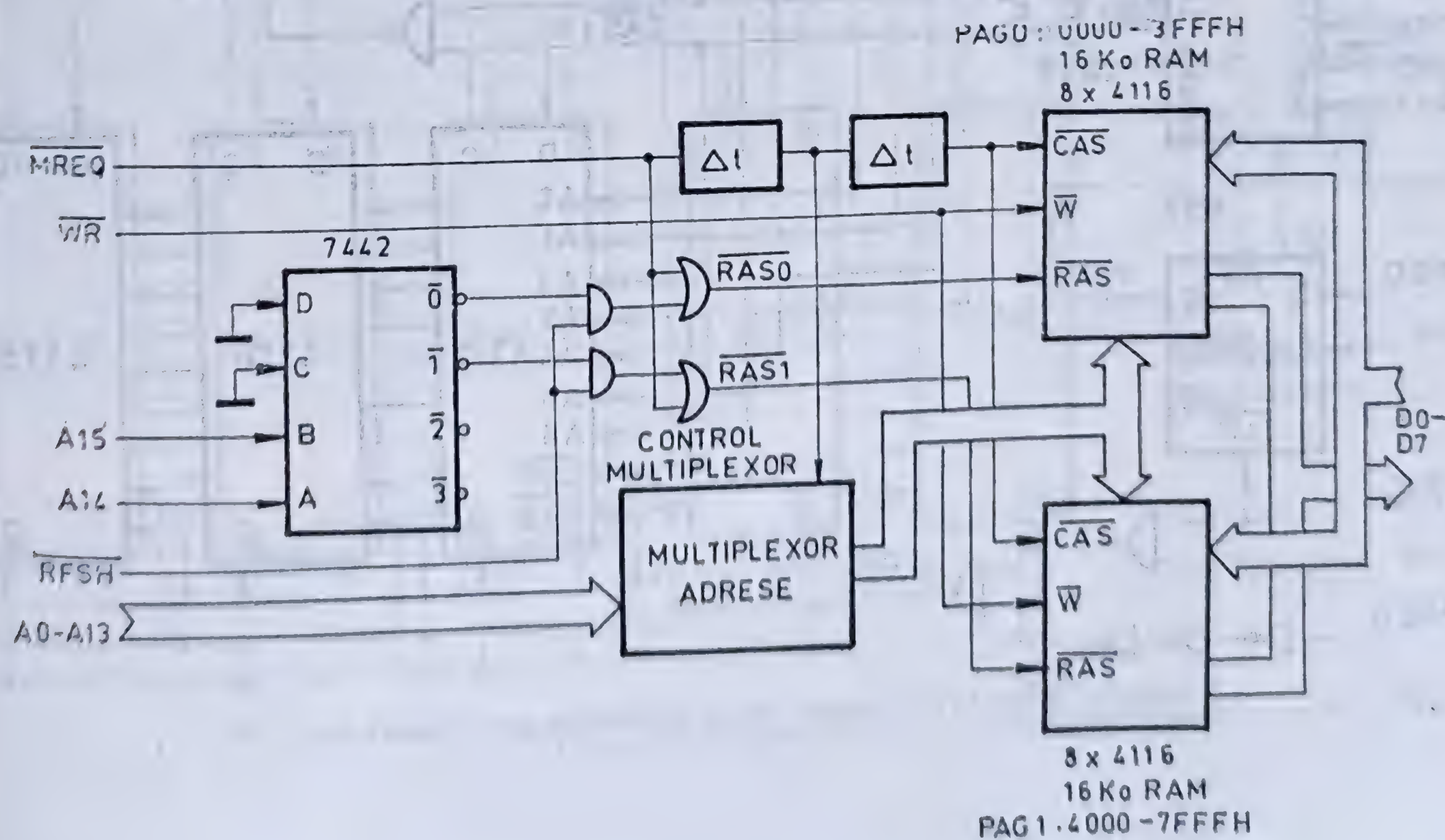
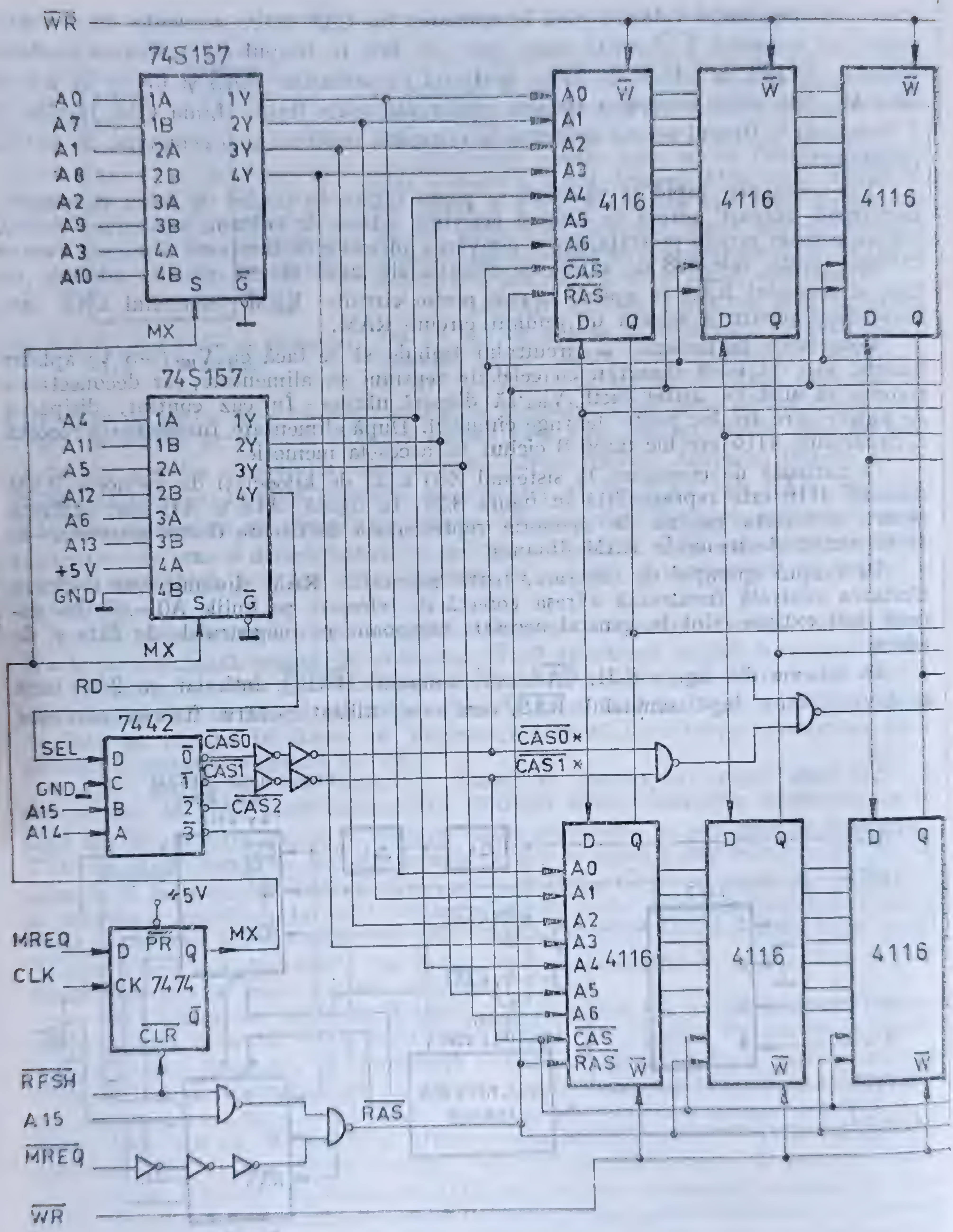


Fig. 8.21 Interfașarea memoriilor RAM dinamic 4116 ( $\overline{\text{CAS}}$  comun).







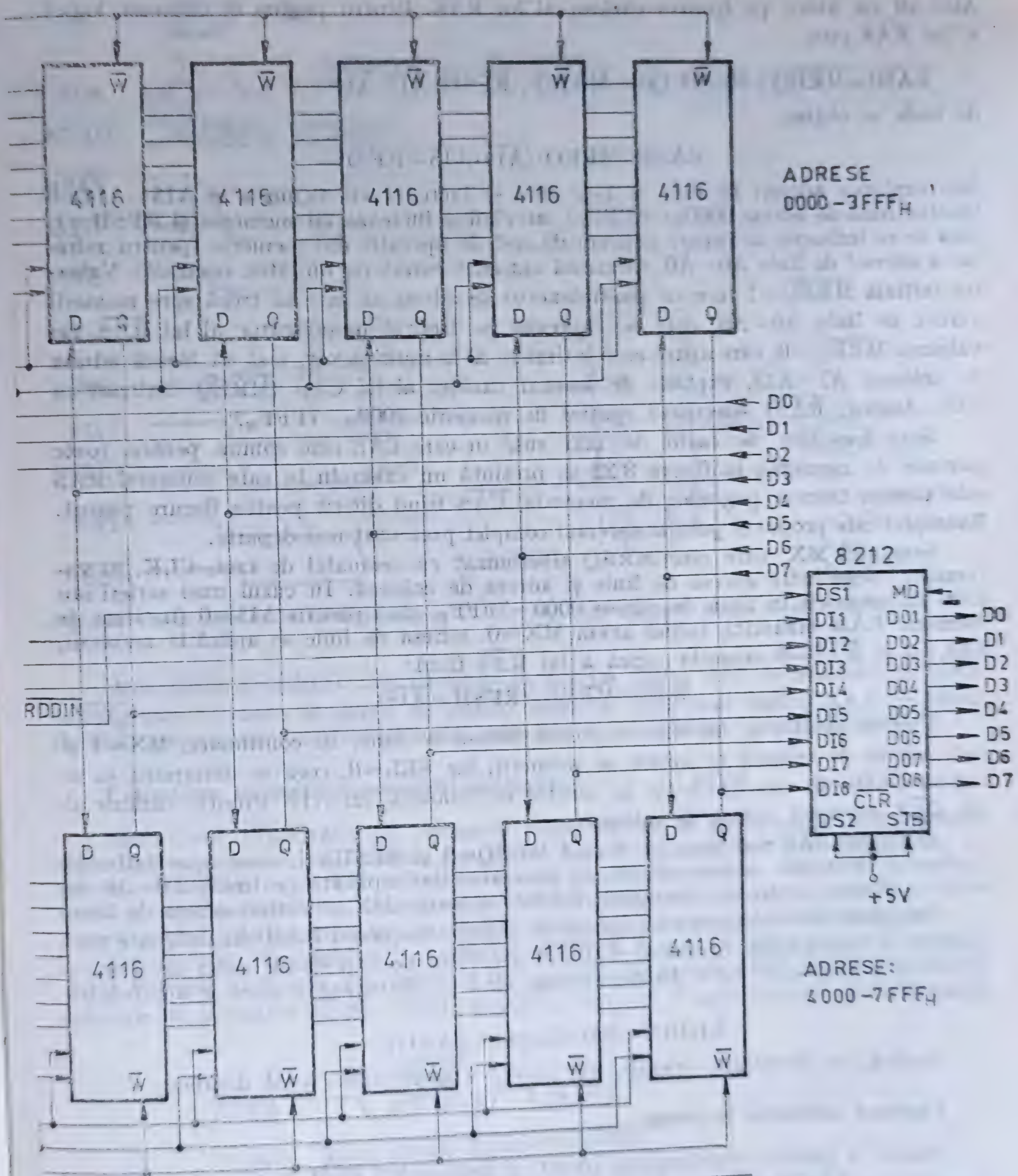


Fig. 8.22 Interfațarea memorii RAM dinamice 4116 (kAS comun).



A0—A6 ale liniei, pe frontul căzător al lui  $\overline{RAS}$ . Pentru pagina 0, expresia logică a lui  $\overline{RAS}$  este

$$\overline{RAS0} = \overline{MREQ} + \overline{RFSH} \cdot \overline{Q0} = \overline{MREQ} + \overline{RFSH} \cdot \overline{A15} \cdot \overline{A14}$$

de unde se obține

$$RAS0 = MREQ \cdot (\overline{A14} \cdot \overline{A15} + RFSH)$$

deci captarea adresei de linie se face când se lucrează cu memoria și  $A15 = A14 = 0$  (pentru zona de adrese  $0000_H - 3FFF_H$ ) sau când se lucrează cu memoria și  $RFSH = 1$ , ceea ce se întâmplă la fiecare aducere de cod de operație din memorie (pentru refreșarea adreselor de linie A0—A6, furnizată automat corect de unitatea centrală). Valoarea inițială  $\overline{MREQ} = 1$  face ca multiplexorul de adrese să lase să treacă spre memorii adresa de linie A0—A6, care se captează pe frontul descrescător al lui  $\overline{RAS}$ , iar valoarea  $\overline{MREQ} = 0$ , care apare cu o întârziere  $\Delta t$  la multiplexor, lasă să treacă adresa de coloană A7—A13, captată de frontul căzător al lui  $\overline{CAS}$  ( $\overline{MREQ}$  întârziat cu  $2\Delta t$ ). Analog,  $\overline{RAS1}$  selectează spațiul de memorie  $4000_H - 7FFF_H$ .

Spre deosebire de cazul de mai sus, în care  $\overline{CAS}$  este comun pentru toate paginile de memorie, în figura 8.22 se prezintă un exemplu în care semnalul  $\overline{RAS}$  este comun tuturor paginilor de memorie,  $\overline{CAS}$  fiind diferit pentru fiecare pagină. Exemplul este propus și pentru sistemul complet prezentat mai departe.

Semnalul MX, care este  $\overline{MREQ}$  sincronizat cu semnalul de tact, CLK, al sistemului, alege între adresa de linie și adresa de coloană. În cazul unei scrieri sau citiri de memorie în zona de adrese  $0000 - 7FFF_H$ , deci pentru  $A15 = 0$  (în zona de memorie RAM dinamic), inițial avem  $MX = 0$ , adresa de linie se aplică la memorii,  $SEL = 1$  și  $\overline{RAS} = 0$  expresia logică a lui  $\overline{RAS}$  fiind:

$$RAS = MREQ \cdot (RFSH + \overline{A15})$$

Frontul căzător al lui  $\overline{RAS}$  captează adresa de linie. În continuare,  $MX = 1$  și deci adresa de coloană se aplică pe memorii, iar  $SEL = 0$ , ceea ce determină să se obțină  $\overline{CAS0} = 0$  sau  $\overline{CAS1} = 0$ , în funcție de valoarea lui A14. Frontul căzător al lui  $\overline{CAS}$  captează adresa de coloană.

Semnalul  $\overline{RAS}$  mai trece la 0 când  $\overline{MREQ} = 1$  și  $RFSH = 1$ , ceea ce se întâmplă în fiecare ciclu M1, adresa corectă de refreșare fiind aplicată pe liniile A0—A6 de către unitatea centrală. Semnalul  $RFSH = 1$  șterge MX, selectând adresa de linie.

Diagrama în timp pentru o operație de citire a memoriei RAM dinamic este prezentată în figura 8.23. Semnalul  $\overline{RDDIN}$  are rolul de a deschide calea de ieșire a datelor din memoriile RAM dinamic numai când se efectuează o citire a lor, funcția logică fiind:

$$RDDIN = RD \cdot (\overline{CAS0} + \overline{CAS1})$$

Analog, se efectuează operația de scriere a memoriilor RAM dinamic.

### Captarea adreselor în sistem

Pentru a garanta funcționarea corectă a memoriilor RAM dinamic, cei doi biți superiori de adresă, A15 și A14, care nu sînt captați de semnalele  $\overline{RAS}$  și  $\overline{CAS}$ , trebuie captați într-un registru, de exemplu cu bistabile, ca în figura 8.24. Acest lucru este necesar deoarece nu este sigur că magistrala de adrese este validă înaintea frontului crescător al lui  $\overline{MREQ}$  la o operație de aducere din memorie a codului unei instrucțiuni. Celelalte linii de adresă se captează intern, în memoriile dinamice.



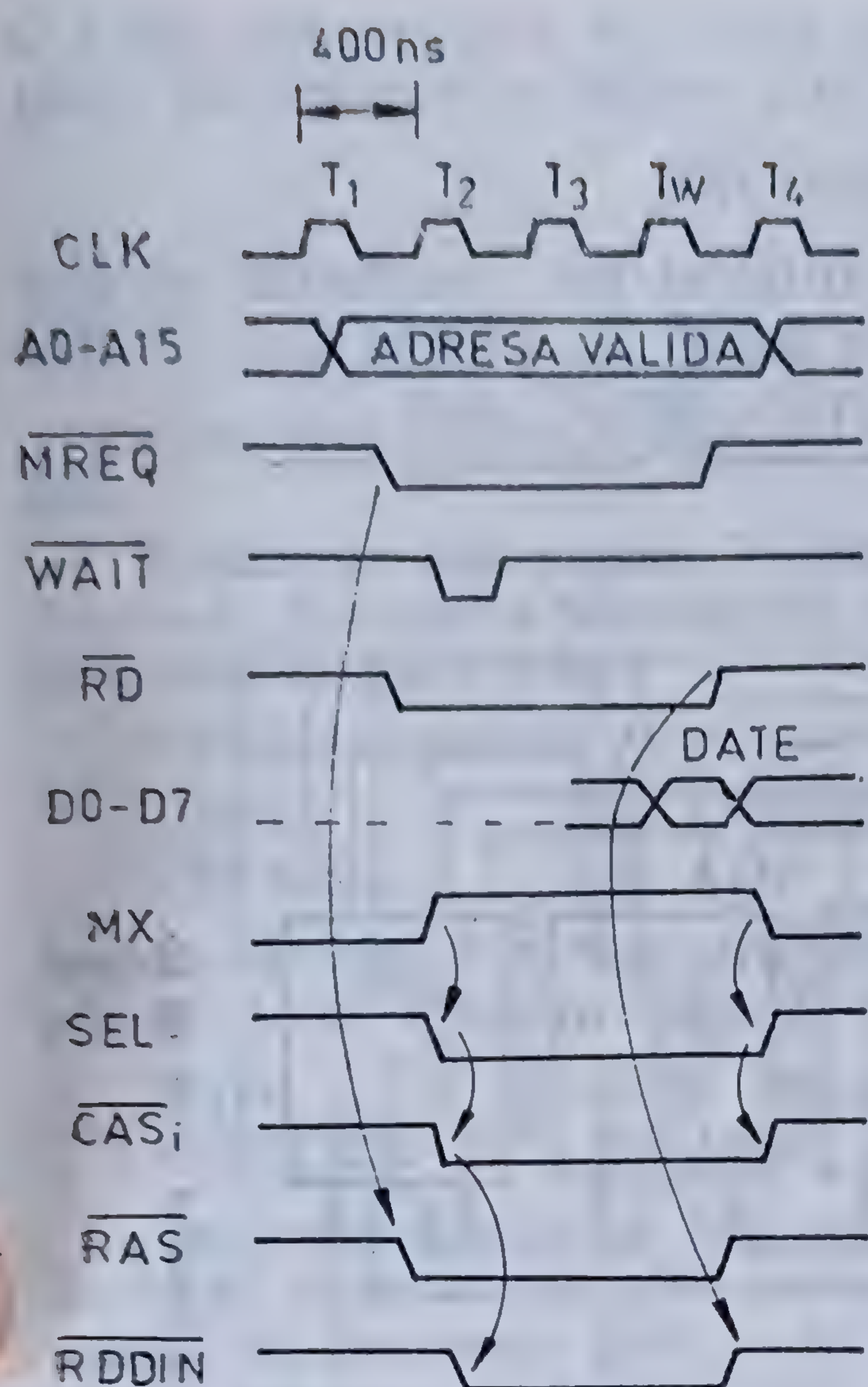


Fig. 8.23 Citirea memoriilor RAM dinamic.

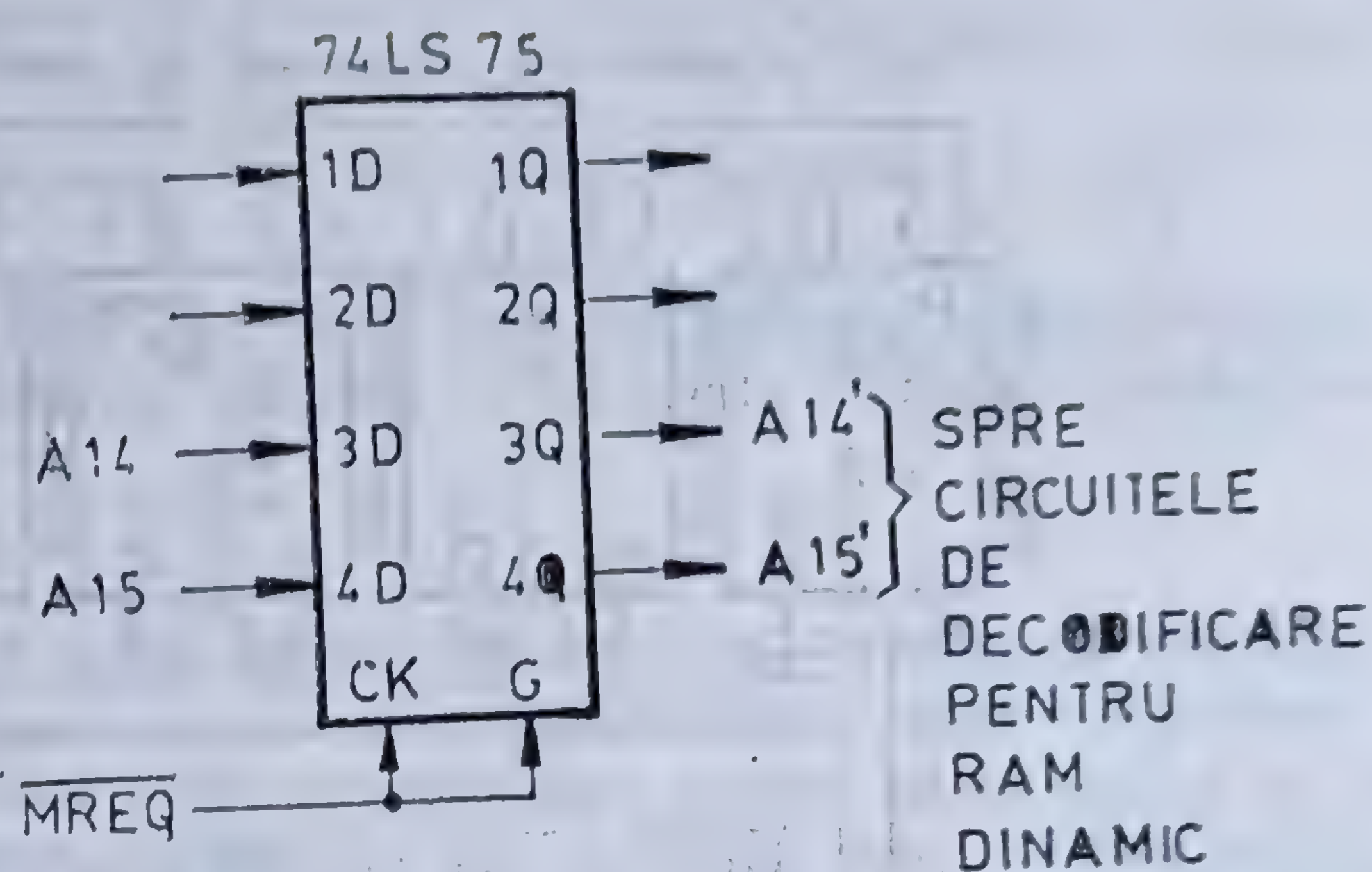


Fig. 8.24 Captarea biților superiori de adresă.

Dacă liniile de adresă care participă la funcția logică  $\overline{RAS}$  se schimbă cît timp  $\overline{MREQ}$  este 0, poate să apară un impuls scurt pe linia (sau liniile)  $\overline{RAS}$ , cu efect de distrugere a unei linii de date din memoria RAM dinamic (figura 8.25).

### Extinderea capacității memoriei sistemului

Extinderea capacității de memorie peste 64 ko se poate realiza cu ajutorul unor linii suplimentare de adresă, controlate de exemplu cu bistabile, considerate în sistem ca dispozitive de I/E. Microprocesorul poate adresa doar 64 ko de memorie, deci extensia memoriei se face introducînd în sistem circuite de memorie care formează pagini diferite fizic, dar în același spațiu de adrese pentru microprocesor.

Un exemplu de „dublare” a primilor 48 ko ai memoriei sistemului, cu memorii RAM dinamic 4116 este prezentat în figura 8.26. Memoriile 4116 din pagina 0 sînt selectate de semnalele  $\overline{CAS0}$ ,  $\overline{CAS1}$ , și  $\overline{CAS2}$ , care pot fi active numai dacă intrarea

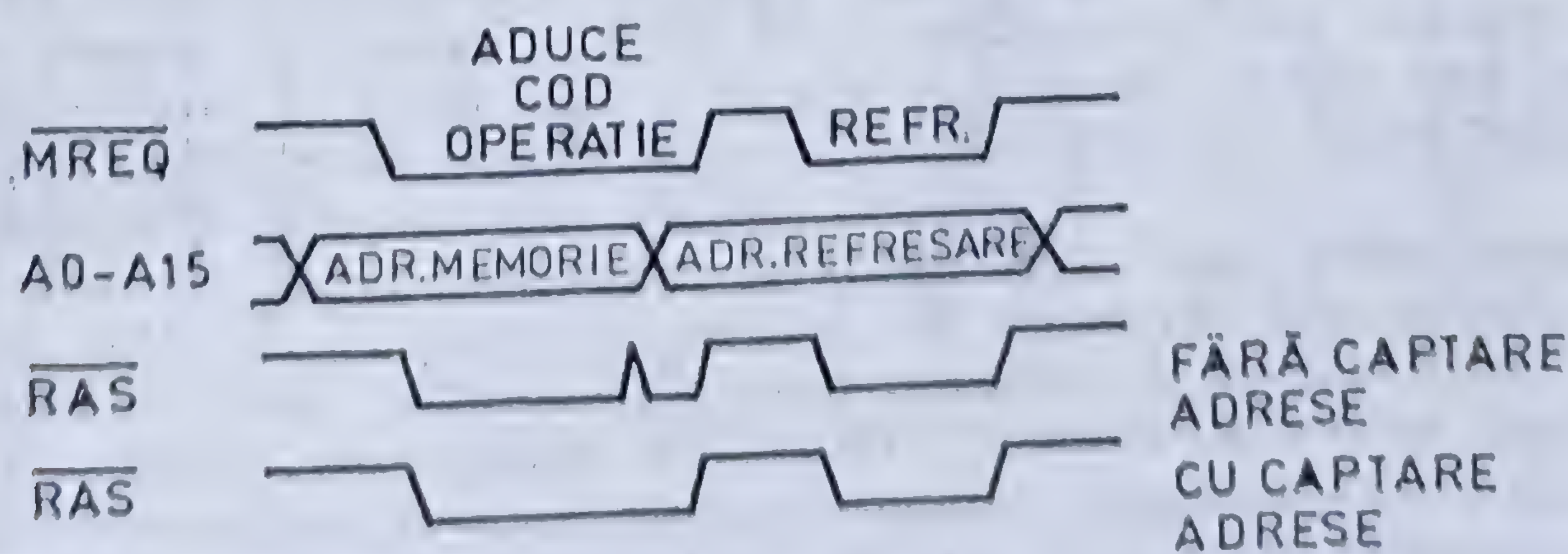


Fig. 8.25 Semnalul  $\overline{RAS}$  cu și fără captarea adreselor.



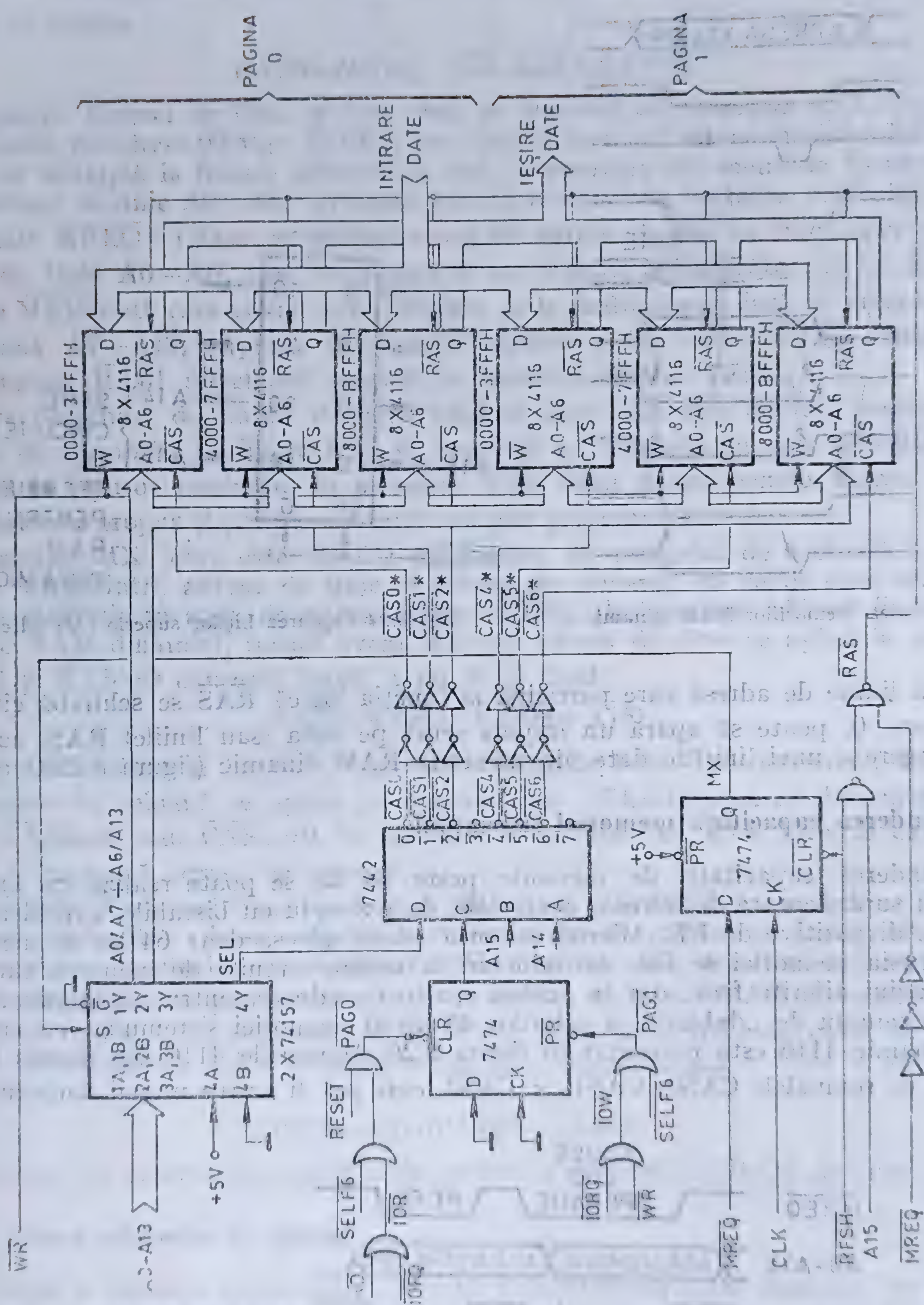


Fig. 8.26 Extinderea capacității memoriei sistemului peste 64 ko.



C a decodicatorului 442, deci ieșirea PAG a bistabilului 474 este la 0 logic. Ștergerea bistabilului se obține prin activarea semnalului.

$$\overline{\text{PAGO}} = \overline{\text{RESET}} \cdot (\overline{\text{SELF6}} + \overline{\text{IOR}})$$

deci la inițializare, sau la citirea portului cu adresa  $\text{F6}_H$  ( $\overline{\text{SELF6}}$  se formează ca și  $\overline{\text{SELF4}}$  sau  $\overline{\text{SELF5}}$  din figura 8.6).

Semnalul  $\overline{\text{IOR}} = \overline{\text{IORQ}} + \overline{\text{RD}}$  este activ numai când se citește un dispozitiv de I/E.

Memoriile din pagina 1 sînt selectate de  $\overline{\text{CAS4}}$ ,  $\overline{\text{CAS5}}$  și  $\overline{\text{CAS6}}$ , când semnalul  $\text{PAG} = 1$ . Înscrierea bistabilului avînd ieșirea PAG se face prin scrierea unui cuvînt la adresa de port  $\text{F6}_H$ .

Utilizarea paginii 0 se poate realiza de exemplu cu instrucțiunile

```
DBF6      IN A,(F6H)
C3XXXX    JP ADR
```

înscrie în spațiul de memorie  $\text{C000}_H - \text{FFFF}_H$ , care nu face parte din zona dublată, adresa ADR fiind în pagina 0. Utilizarea paginii 1 se poate realiza asemănător:

```
D3F6      OUT (F6H),A
C3XXXX    JP ADR
```

În instrucțiunile de mai sus, ADR este adresa care trebuie atinsă în pagina 1. În locul instrucțiunii JP poate să apară orice altă instrucțiune care se referă la spațiul de memorie  $\text{0000}_H - \text{BFFF}_H$ . Această instrucțiune poate să apară și la un număr oarecare de instrucțiuni după instrucțiunile  $\text{IN A,(F6H)}$  sau  $\text{OUT (F6H),A}$ . Asemănător, se poate dubla și zona de memorie cu adresele  $\text{C000}_H - \text{FFFF}_H$ . Schimbarea paginii trebuie făcută cu instrucțiuni aflate într-o zonă de memorie care nu este afectată de schimbare. Pe același principiu, se poate multiplica de mai multe ori același spațiu de adrese. Atingerea uneia sau alteia dintre paginile suprapuse de memorie se poate realiza cu instrucțiuni de I/E, ca mai sus, sau chiar cu instrucțiuni de citire/scriere a unor locații de memorie, care nu se vor mai utiliza în acest caz pentru operații obișnuite de depozitare de date.

## 8.2 APLICAȚII

### Conectarea unui convertor numeric analog

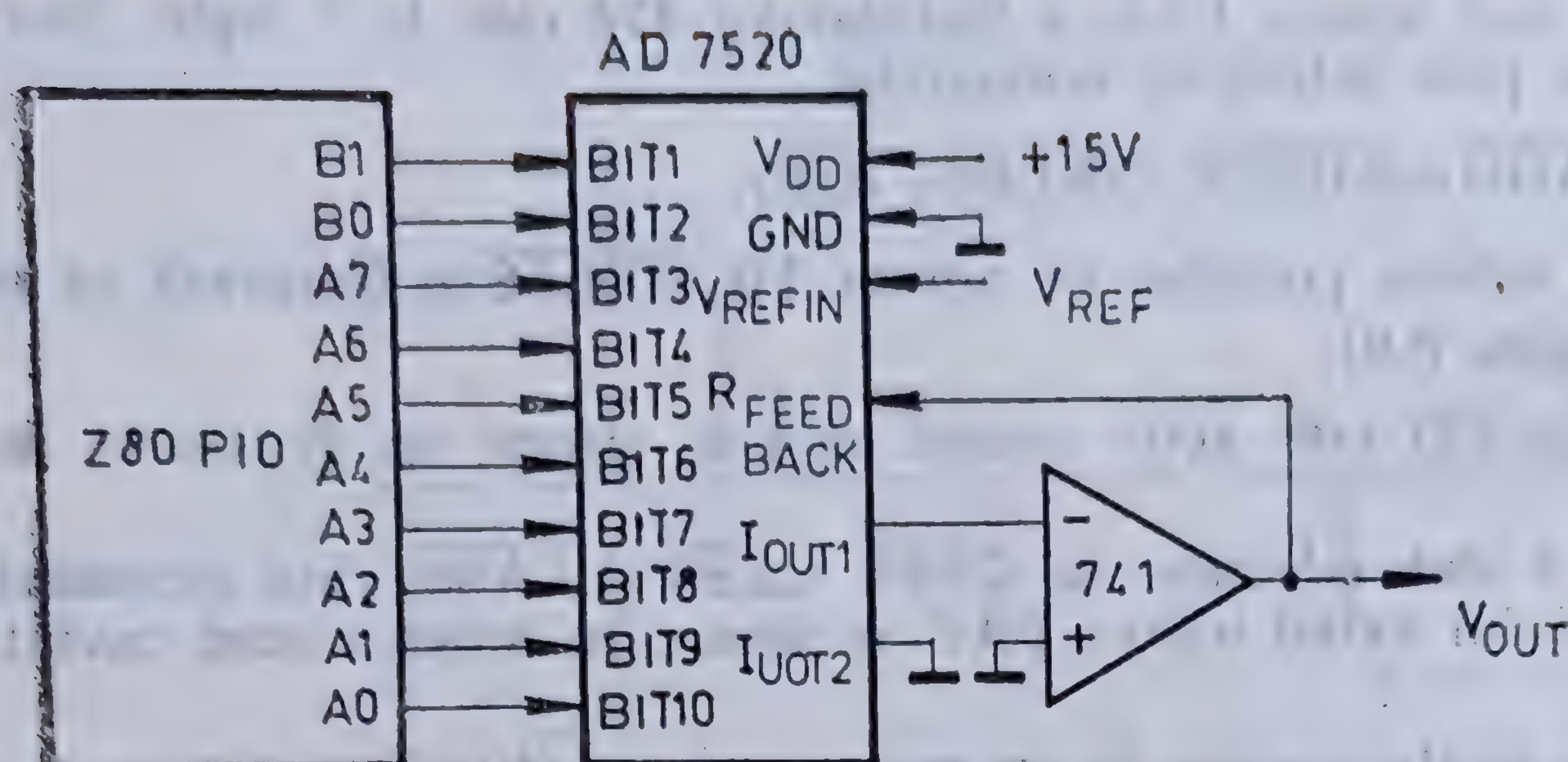
Se va prezenta ca exemplu conectarea convertorului numeric-analog AD7520 (K572PA1A), de 10 biți. Funcționează cu o tensiune de alimentare în domeniul  $+5\text{ V} \div +15\text{ V}$  și are aplicații în multiplicarea numeric/analogică, generarea de caractere cu tuburile cu raze catodice, în sursele de alimentare programabile, în circuitele de amplificare cu factor de amplificare controlat numeric etc.

Modul de funcționare binar unipolar (multiplicare în două cadrane) al convertorului este ilustrat în figura 8.27, în care convertorul este conectat la un circuit Z80 PIO. Conectarea se poate face și la alte registre de 10 biți (sau mai puțin), sau de exemplu, la ieșirile unor numărătoare. Circuitul PIO din figura 8.27 poate fi programat în modul 3 (intrare/ieșire de bit).

Bitul 1 este cel mai semnificativ, iar bitul 10 cel mai puțin semnificativ. Tensiunea de referință,  $V_{\text{REF}}$ , poate fi pozitivă sau negativă, deci circuitul este capabil de multiplicare în două cadrane. Valoarea tensiunii la ieșirea  $V_{\text{OUT}}$ , în funcție de codul numeric aplicat pe intrările  $\text{BIT1} - \text{BIT10}$  apare în aceeași figură.

Variația de tensiune obținută la ieșire pentru modificare de 1 bit (corespunzătoare celui mai puțin semnificativ bit) este de  $V_{\text{REF}} \cdot 2^{-10}$ . Tensiunea de referință trebuie să se găsească în domeniul  $-10\text{ V} \div +10\text{ V}$ .





INTRĂRI NUMERICE										IEȘIRE ANALOGICĂ	
BIT	1	2	3	4	5	6	7	8	9	10	$V_{OUT}$
	1	1	1	1	1	1	1	1	1	1	$-V_{REF} (1 - 2^{-10})$
	1	0	0	0	0	0	0	0	0	1	$-V_{REF} (\frac{1}{2} + 2^{-10})$
	1	0	0	0	0	0	0	0	0	0	$-V_{REF} \cdot \frac{1}{2}$
	0	1	1	1	1	1	1	1	1	1	$-V_{REF} (\frac{1}{2} - 2^{-10})$
	0	0	0	0	0	0	0	0	0	1	$-V_{REF} \cdot 2^{-10}$
	0	0	0	0	0	0	0	0	0	0	0

Fig. 8.27 Funcționarea binară unipolară (multiplicarea în 2 cadrane) a circuitului AD7520.

Reglarea valorii 0 la ieșirea amplificatorului operațional se face cu toate intrările digitale la masă (se reglează  $0 V \pm 1 mV$  la  $V_{OUT}$ ). Reglarea amplificării se realizează conectând toate intrările digitale ale convertorului la sursa de  $+15 V$  și introducând un rezistor  $R$  ( $0 \div 500 \Omega$ ) între ieșirea  $V_{OUT}$  a amplificatorului și intrarea  $R_{FEEDBACK}$  a convertorului pentru a mări  $V_{OUT}$ , sau un rezistor  $R$  în serie cu  $V_{REF}$  ( $0 \div 500 \Omega$ ).

Funcționarea bipolară este ilustrată în figura 8.28, și este posibilă deoarece intrările numerice acceptă numere bipolare, iar  $V_{REF}$  acceptă o intrare analogică bipolară. Se obține astfel o funcție de multiplicare în 4 cadrane.

Codificarea la intrare este în complement față de 2, modificat, așa cum reiese din aceeași

figură. Modificarea numărului de la intrare cu 1 bit (cel mai puțin semnificativ bit) determină la ieșire o variație a tensiunii  $V_{OUT}$  cu  $V_{REF} \cdot 2^{-9}$ .

Reglarea decalajului se realizează în următoarea ordine: se conectează  $V_{REF}$  la  $+10 V$ , intrările numerice la  $+15 V$  (1 logic), se ajustează ieșirea amplificatorului 2 la  $0 V \pm 1 mV$ ; se lasă BIT1 la  $+15 V$ , se conectează celelalte intrări numerice la  $0 V$ , și se ajustează ieșirea amplificatorului 1 la  $0 V \pm 1 mV$  ( $V_{OUT}$ ).

Reglarea amplificării se face asemănător cu reglarea din cazul funcționării unipolare.

O altă posibilitate de utilizare este divizarea analog/numerică. În cazul funcționării unipolare (figura 8.29), funcția de transfer este

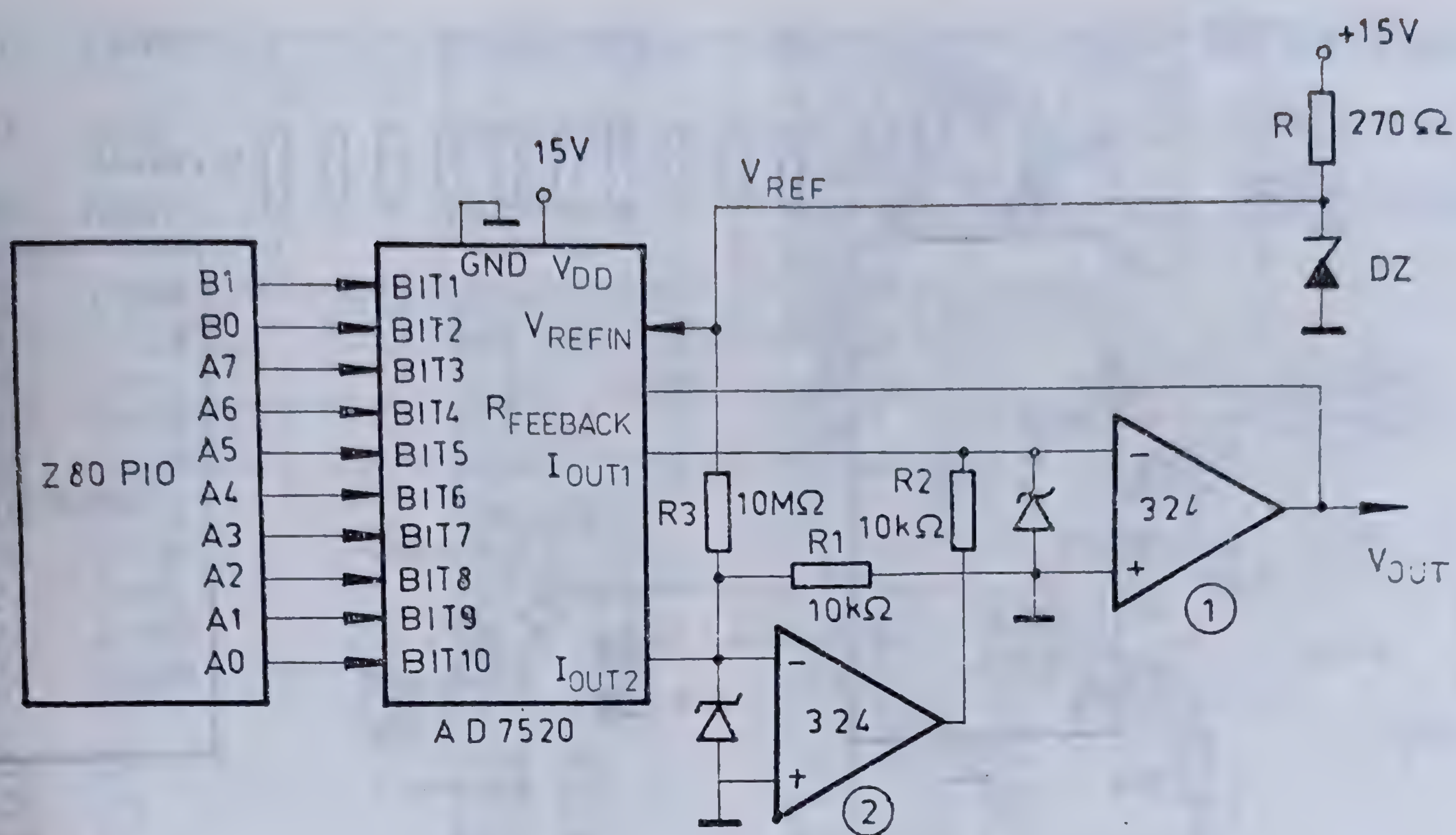
$$V_o = -V_{IN} \left( \frac{A_1}{2^1} + \frac{A_2}{2^2} + \dots + \frac{A_n}{2^n} \right)$$

unde coeficienții  $A_i$  au valoarea 1 sau 0, în conformitate cu valoarea bitului corespunzător la intrările numerice. Conectând convertorul pe calea de reacție a unui amplificator operațional, ca în figura 8.29, funcția de transfer devine

$$V_o = \frac{-V_{IN}}{\frac{A_1}{2^1} + \frac{A_2}{2^2} + \dots + \frac{A_n}{2^n}}$$

ceea ce înseamnă divizarea intrării analogice  $V_{IN}$ . Cu toți biții  $A_i=0$ , amplificatorul se saturează la limită, împărțirea cu 0 nefiind definită. Cu  $A_n=1$  ( $A_{10}=1$ ), amplificarea este de  $2^{10}=1024$ . Cu toți biții  $A_i=1$ , amplificarea este egală cu 1 ( $\pm 1/2^{10}$ ).





INTRARI NUMERICE										IESIRE ANALOGICA	
BIT 1	2	3	4	5	6	7	8	9	10	V <sub>OUT</sub>	
1	1	1	1	1	1	1	1	1	1	$-V_{REF} (1 - 2^{-9})$	
1	0	0	0	0	0	0	0	0	1	$-V_{REF} 2^{-9}$	
1	0	0	0	0	0	0	0	0	0	0	
0	1	1	1	1	1	1	1	1	1	$+V_{REF} 2^{-9}$	
0	0	0	0	0	0	0	0	0	1	$+V_{REF} (1 - 2^{-9})$	
0	0	0	0	0	0	0	0	0	0	$V_{REF}$	

Fig. 8.28 Funcționarea bipolară (multiplicarea în 4 cadrane) a circuitului AD7520.

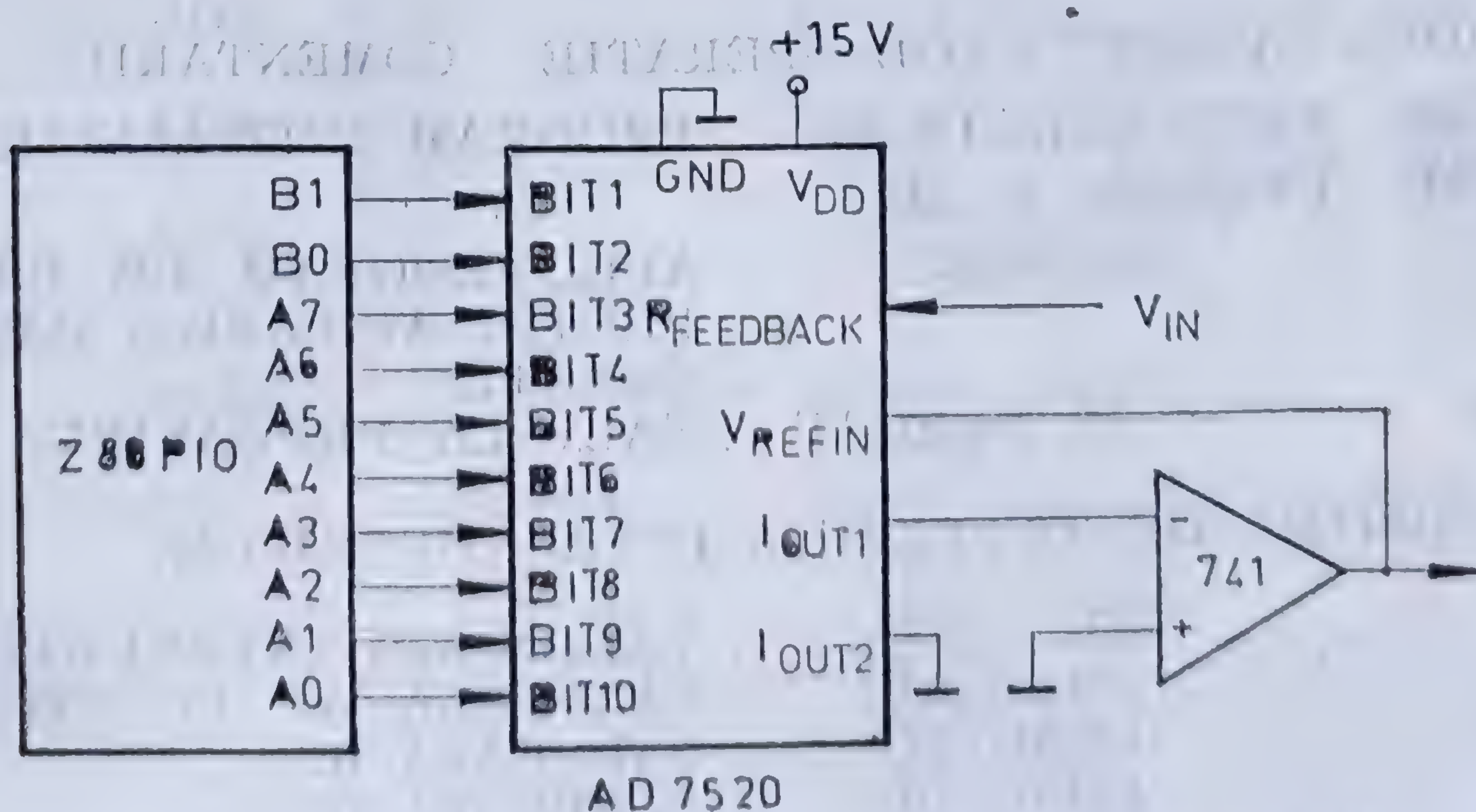


Fig. 8.29 Divizarea analog/numerică realizată cu circuitul AD7520.



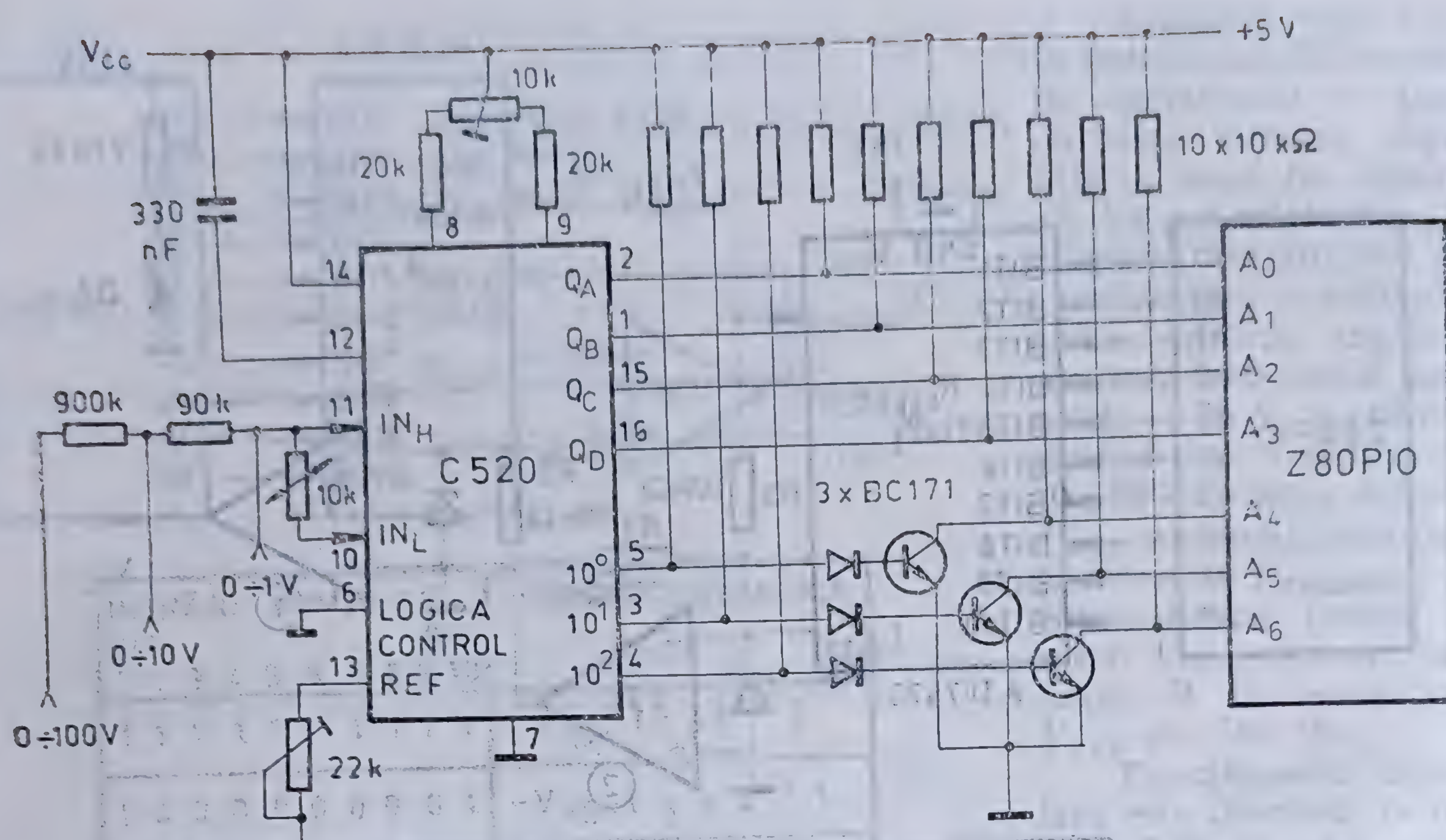


Fig. 8.30 Conectarea unui convertor analog-numeric C520 la circuitul Z80 PIO.

### Conectarea unui convertor analog-numeric

Un convertor analog-numeric utilizat relativ frecvent este C520, care transformă tensiunea de intrare într-un cod BCD de 3 cifre, multiplexate pe ieșirile  $Q_D$ ,  $Q_C$ ,  $Q_B$ ,  $Q_A$ , indicarea ordinului de mărime (unități, zeci, sute) fiind efectuată de 3 ieșiri ale convertorului:  $10^0$ ,  $10^1$  și  $10^2$ .

Conectarea la circuitul PIO, programat în modul 3 (intrare/ieșire de bit) este reprezentată în figura 8.30. Programul de utilizare a circuitului convertor, conectat la canalul A al circuitului Z80 PIO este listat în continuare. Adresele sînt: DAT =  $85_H$  pentru portul A, date; CTR =  $87_H$ , pentru portul A, control; IST =  $0D32_H$  pentru tabelul cu adresele subrutinelor de întrerupere; MEM =  $0D40_H$ , adresa de bază a zonei de memorie pentru depunerea valorilor măsurate; DIS =  $0C15_H$ , adresa memoriei de vizualizare.

### PROGRAMUL PRINCIPAL

ADRESA	CODUL	ETICHETA	COD OPERAȚIE	COMENTARIU
2000	CD4C20	ADU:	CALL INIPIO	PROGRAM INITIALIZARE PIO
2003	CD6720	PREGAF:	CALL BCDASC	APEL PROGRAM DE PREGATIRE PENTRU AFISARE A DATELOR MĂ- SURATE
2006	18FB		JR PREGAF	SALT LA PREGATIREA AFISARII

### SUBROUTINA DE TRATARE A INTRERUPERILOR

2008	FB	EI		VALIDEAZA INTRERUPERILE
2009	F5	PUSH AF		SALVEAZA AF PE STIVA
200A	C5	PUSH BC		SALVEAZA BC
200B	D5	PUSH DE		SALVEAZA DE
200C	E5	PUSH HL		SALVEAZA HL



200D	11400D		LD DE, MEM	; INCARCA ADRESA DE DEPUERERE
2010	0602		LD B, 02H	; A DATELOR IN DE
2012	0E87		LD C, CTR	; NUMARUL NOILOR CUVINTE DE
2014	DB85	IN1:	IN A, DAT	; COMANDA, IN B
2016	CB77		BIT 6, A	; ADRESA PORTULUI DE CONTROL
2018	2015		JRNZ SUTE	; IN C
201A	CB67		BIT 4, A	; CITESTE DATE
201C	2022		JRNZ UNIT	; SINT SUTE?
201E	CB6F		BIT 5, A	; DA, SALT LA ADRESA SUTE
2020	2014		JRNZ ZECI	; SINT UNITATI?
2022	18F0		JR IN1	; DA, SALT LA ADRESA UNIT
2024	DB85	IN2:	IN A, DAT	; SINT ZECI?
2026	E60F		AND 0FH	; DA, SALT LA ADRESA ZECI
2028	12		LD (DE), A	; ASTEAPTA CIFRA BCD
2029	E1		POP HL	; CITESTE DATE
202A	D1		POP DE	; SELECTEAZA CIFRA BCD
202B	C1		POP BC	; DEPUNE IN MEMORIE
202C	F1		POP AF	; REFACE HL
202D	ED4D		RETI	; REFACE DE
202F	214A20	SUTE:	LD HL, CDU	; REFACE BC
2032	EDB3	S1:	OTIR	; REFACE AF
2034	18EE		JR IN2	; REVINE DIN INTRERUPERE
2036	EB	ZECI:	EX DE, HL	; ADRESA CUVINTELOR DE
2037	CBFE		SET 7, (HL)	; COMANDA PENTRU UNITATI
2039	EB		EX DE, HL	; INSCRIE CUVINTELE DE COMAN-
203A	214620		LD HL, CDS	; DA
203D	13	Z1:	INC DE	; SALT LA CITIRE DATE
203E	18F2		JR S1	; INSCRIE BIT INDICATOR
2040	214820	UNIT:	LD HL, CDZ	; ADRESA CUVINTELOR DE COMAN-
2043	13		INC DE	; DA PENTRU SUTE
2044	18F7		JR Z1	; ADRESA CUVINTELOR DE CO-
2046	F7	CDS:	F7H	; MANDA PENTRU ZECI
2047	BF		BFH	; DOUA CUVINTE DE COMANDA
2048	F7	CDZ:	F7H	; PENTRU SUTE
2049	EF		EFH	; DOUA CUVINTE DE COMANDA
204A	F7	CDU:	F7H	; PENTRU ACTIVARE ZECI
204B	DF		DFH	; DOUA CUVINTE DE COMANDA
				; PENTRU ACTIVARE UNITATI

#### SUBROUTINA PENTRU INITIALIZARE PIO

204C	3E0D	INIPIO:	LD A, ISTH	; OCTET SUPERIOR DIN IST, IN A
204E	ED47		LD I, A	; OCTETUL IN REGISTRUL DE
2050	210820		LD HL, INT	; INTRERUPERI
2053	22320D		LD (IST), HL	; ADRESA DE INCEPUT A PROGRA-
				; MULUI DE INTRERUPERE, IN HL
				; DEPUNE IN TABELUL CU ADRESE-
				; LE SUBROUTINELOR DE INTRERU-
				; PERE



2056	216220	LD HL,CVCD	; ADRESA DE BAZA LISTA CUVINTE
			; DE COMANDA PENTRU PIO
2059	0605	LD B,05H	; NUMARUL CUVINTELOR DE CO-
			; MANDA
205B	0E87	LD C,CTR	; ADRESA PORTULUI DE CONTROL,
			; IN C
205D	F3	DI	; DEZACTIVEAZA INTRERUPERILE
205E	EDB3	OTIR	; INSCRIE CUVINTE DE COMANDA
			; IN PIO
2060	FB	EI	; ACTIVEZA INTRERUPERILE
2061	C9	RET	; REVINE DIN SUBROUTINA
2062	32	CVCD: ISTL	; OCTET INFERIOR DIN IST
2063	CF	CFH	; CUVINT COMANDA MOD 3
2064	7F	7FH	; BITUL 7 IESIRE,BITII 6-0 IN-
			; TRARI
2065	F7	F7H	; COMANDA INTRERUPERILOR (EI,
			; AND, HIGH, MASK)
2066	BF	BFH	; MASCA LA INTRERUPERI: BITUL 6
			; GENEREAZA INTRERUPEREA;
			; BFH=10111111B

SUBROUTINA DE PREGATIRE PENTRU AFISARE LA CONSOLA SERIALA,  
A DATELOR CITITE

2067	21400D	BCDASC: LD HL, MEM	; ADRESA VALORILOR MASURATE
206A	11150C	LD DE, DIS	; ADRESA VALORILOR PREGATITE
			; PENTRU AFISARE
206D	0603	LD B,03H	; NUMARUL CIFRELOR
206F	CB7E	BIT 7,(HL)	; TESTEAZA BITUL INDICATOR, IN Z
2071	C8	RZ	; REVINE, DACA (HL)7=0
2072	CBBE	RES 7,(HL)	; (HL)7=0, DACA A FOST 1
2074	7E	DA1: LD A,(HL)	; CIFRA BCD IN A
2075	FE0A	CMP 0AH	; DEPASIRE NEGATIVA?
2077	2004	JRNZ DA2	; DA, CORECTIE DE CIFRA; NU, SALT
			; LA ADRESA DA2
2079	3E2D	LD A,2DH	; CODUL ASCII PENTRU „-”, IN A
207B	1808	JR DA4	;
207D	FE0B	DA2: CMP0BH	; DEPASIRE POZITIVA?
207F	2004	JRNZ DA3	; DA, CORECTIE DE CIFRA; NU, SALT
			; LA ADRESA DA3
2081	3E2B	LD A,2BH	; CODUL ASCII PENTRU „+”, IN A
2083	1802	JR DA4	;
2085	F630	DA3: OR 30H	; SAU LOGIC CU CODUL ASCII
			; DE BAZA PENTRU CIFRE
2087	12	DA4: LD (DE),A	;
2088	23	INC HL	; URMATOAREA CIFRA MASURATA
2089	13	INC DE	; URMATOARE CIFRA DE AFISAT
208A	10E8	DJNZ DA1	; SALT LA DA1 DACA NU S-AU PRE-
			; GATIT TOATE 3 CIFRELE, PENTRU
			; B≠0
208C	C9	RET	; REVINE, PENTRU B=0



## Funcționarea pas cu pas a microprocesorului Z80, în regim de „cite un ciclu de mașină”

Funcționarea pas cu pas a microprocesorului Z80 poate fi realizată în regim „cite un ciclu de mașină”, menținând linia  $\overline{\text{WAIT}}$  a microprocesorului la 0 logic și aplicând un impuls 1 logic de durata perioadei de tact a sistemului pentru fiecare ciclu de instrucțiune care trebuie executat. Circuitul prezentat în continuare are posibilitatea de a opri programul în curs de rulare la un moment oarecare și de a-l executa în continuare pas cu pas, sau de a opri programul în curs de rulare la o adresă fixă, de stop, prestabilită, și de a-l executa în continuare pas cu pas.

Circuitul de coincidență dintre adresa de stop C15—C0, fixată cu ajutorul 16 comutatoare (tip microswitch) și adresa din sistem, A15—A0, poate fi realizat cu ajutorul unor porți de coincidență (SAU EXCLUSIV negat), sau al unor comparatoare, de exemplu 7485, ca în figura 8.31. Compararea se face numai când se citește memoria sistemului, deci când este activ semnalul:

$$\text{MEMR} = \text{MREQ} \cdot \text{RD} = \overline{\text{MREQ}} + \overline{\text{RD}}$$

Circuitul va genera un semnal  $\overline{\text{EQU}} = 0$  când adresa din sistem, în timpul citirii unui octet din memorie, coincide cu adresa de stop, determinând astfel trecerea semnalului  $\overline{\text{WAIT}}$  de la 1 la 0 și a microprocesorului în stare de așteptare ( $\overline{\text{WAIT}}$ ). De notat că în această stare, semnalele de reîmproșare a conținutului memoriilor RAM dinamic nu se mai generează, deci conținutul lor se pierde.

Circuitul de generare a semnalului  $\overline{\text{WAIT}}^*$ , aplicat pe intrarea  $\overline{\text{WAIT}}$  a unității centrale, este prezentat în figura 8.32. Bistabilul B1 este șters de semnalul  $\overline{\text{EQU}}$ , generând semnalul  $\text{STOP} = 1$ , care va determina trecerea microprocesorului în starea de așteptare ( $\overline{\text{WAIT}}^* = 0$ ). Înscrierea acestui bistabil se realizează fie la inițializare ( $\overline{\text{RESET}} = 0$ ) fie la activarea butonului „START”, care generează un monoimpuls  $\text{ST} = 0$  între primele 2 fronturi căzătoare ale semnalului de tact CLK care urmează acționării butonului, cu ajutorul generatorului de monoimpuls format din bistabilele B2 și B3. Rolul butonului „START” este de a permite rularea în viteză a unui segment de program pînă la o nouă adresă de stop, după oprirea pe o adresă de stop fixată anterior.

Comutatorul „STOP PE ADRESA” permite rularea în viteză a programului pînă la atingerea adresei de stop, când se trece în regim pas cu pas. Semnalul care realizează această operație este SA.

Comutatorul „AUTOMAT” deschis face ca  $\overline{\text{WAIT}}^* = 1$ , deci programul este rulat în viteză. Microprocesorul poate fi trecut în starea de așteptare și de semnalul  $\overline{\text{WAITM}}$  de la memorii mai lente, care nu pot funcționa la viteza maximă a microprocesorului.

Comutatorul „PAS CU PAS” deschis, când primele 2 sînt închise, determină  $\overline{\text{WAIT}}^* = 0$ , dacă nu se acționează butonul „1 PAS” și  $\overline{\text{WAIT}}^* = 1$  timp de o perioadă de tact, dacă se acționează butonul „1 PAS”. Semnalul  $\overline{\text{WAIT}}^*$  trece la 1 ca efect al apariției semnalului  $\text{UP}^* = 1$ , între primele 2 fronturi crescătoare de tact care urmează acționării butonului „1 PAS”. Poarta trigger Schmitt 413 are rolul de a forma frontul semnalului UP, aplicat pe intrarea generatorului de monoimpuls format din bistabilele B4 și B5.

Expresia logică a semnalului  $\overline{\text{WAIT}}^*$  este:

$$\overline{\text{WAIT}}^* = (\text{AUT} + \text{SA} \cdot \text{STOP}) \cdot \overline{\text{WAITM}} + (\text{PP} + \text{SA} \cdot \text{STOP}) \cdot \text{UP}^*$$

Se observă că dacă nu se acționează butonul „1 PAS”, deci  $\text{UP}^* = 0$ , iar  $\overline{\text{WAITM}} = 0$ , atunci  $\overline{\text{WAIT}}^* = 0$  și microprocesorul trece în starea de așteptare. Dacă  $\overline{\text{WAITM}} = 1$ , atunci combinațiile utile ale semnalelor SA, AUT, PP, determinate de cele 3 comutatoare din figura 8.32, ca și valoarea logică a lui  $\overline{\text{WAIT}}^*$ , acțiunea și utilizarea lor sînt cele din figura 8.33.



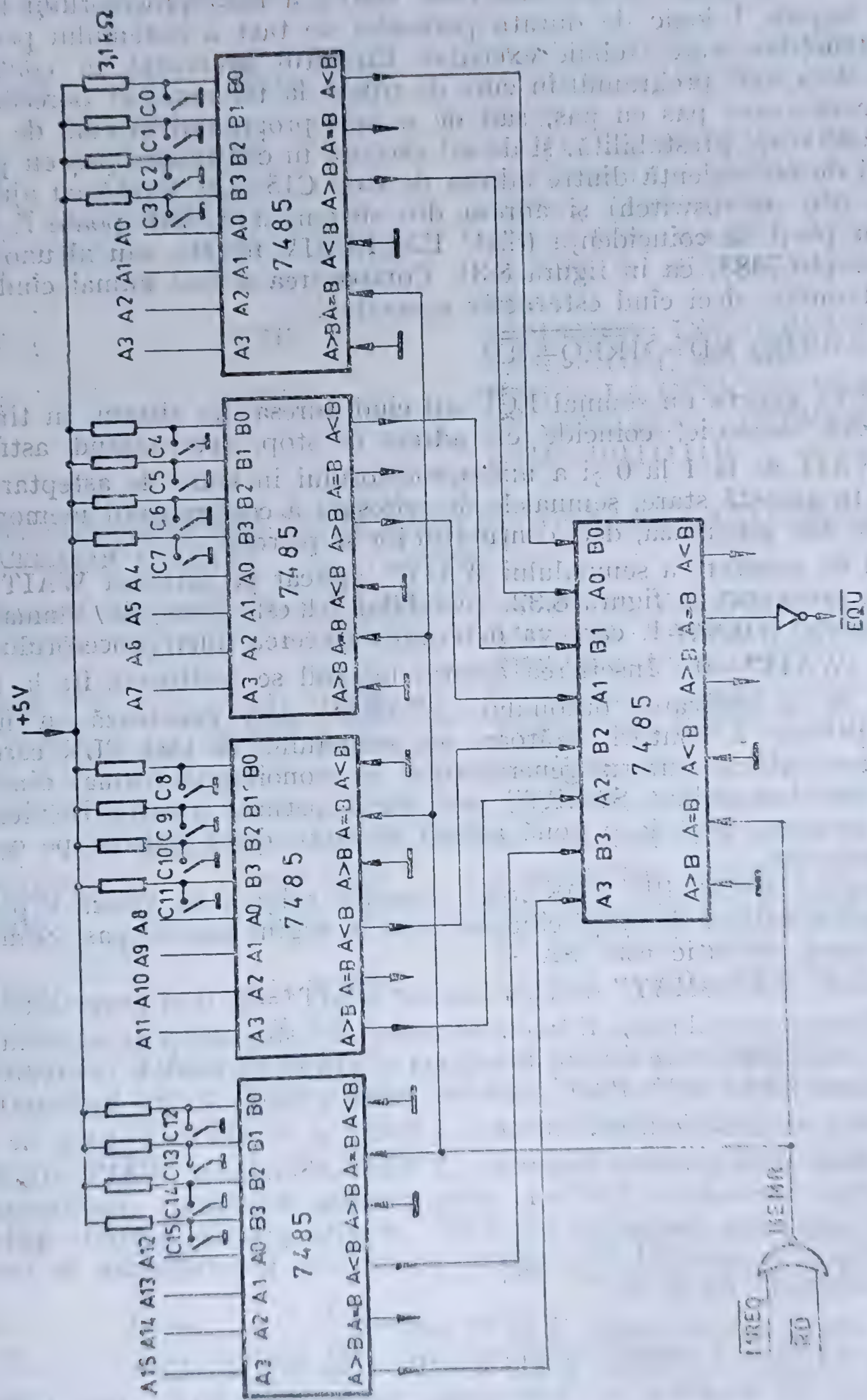


Fig. 8.31 Circuit de comparare a adreselor de stop cu adresa din sistem.







AUT	SA	PP	$\overline{\text{WAIT}}^*$	ACȚIUNEA
0	0	0	0	CPU ÎN STARE DE AȘTEPTARE, $\overline{\text{WAIT}}^* = 0$
1	0	0	1	CPU RULEAZĂ PROGRAMUL, $\overline{\text{WAIT}}^* = 1$
0	1	0	$\overline{\text{STOP}}^* \text{ STOP}^*$ $\text{UP}^*$	$\overline{\text{WAIT}}^* = 1$ PÎNĂ LA ATINGEREA ADRESEI DE STOP, DUPĂ CARE $\overline{\text{WAIT}}^* = \text{UP}^*$
0	0	1	$\text{UP}^*$	CPU EXECUTĂ UN CICLU DE MASINĂ CÎND $\text{UP}^* = 1$

Fig. 8.33 Controlul semnalului  $\overline{\text{WAIT}}$ .

Prima combinație este utilizată ca poziție de trecere între următoarele: funcționare automată, funcționare automată pînă la adresa de stop, iar în continuare pas cu pas și respectiv funcționare pas cu pas (cîte un ciclu de mașină).

De notat că un ciclu de mașină reprezintă o subdiviziune a unui ciclu de instrucțiune și este un multiplu al perioadei de tact a sistemului, numită stare. Un ciclu de instrucțiune (timpul necesar efectuării tuturor operațiilor pentru execuția unei instrucțiuni) este format din 1, 2, 3, 4, 5 sau 6 cicluri de mașină, iar un ciclu de mașină poate fi format din 3, 4 sau 5 stări, dacă nu se adaugă stări adiționale de așteptare (WAIT).

Celelalte combinații posibile ale semnalelor AUT, SA, PP nu duc la situații noi (se repetă unele dintre cele prezentate) și pot fi neutilizate.

Afișarea datelor și adreselor din sistem în scopul citirii lor în regim pas cu pas poate fi realizată sub formă de cifre hexazecimale, cu ajutorul unor circuite de afișare cu 7 segmente MDE 2101, comandate cu memorii PROM 74188, înscrise astfel încît pentru o combinație de 4 biți la intrare să realizeze „aprinderea” led-urilor corespunzătoare cifrei hexazecimale pe care o reprezintă cei 4 biți.

Formatul cifrelor hexazecimale ales este prezentat în figura 8.34, iar conținutul memoriei PROM, din care se utilizează doar 16 locații din 32, este cel din figura 8.35.

Aprinderea unui led va avea loc dacă ieșirea memoriei PROM care îl comandă va fi la 0 logic. Dintre intrările de adresă E, D, C, B, A ale memoriei, se utilizează doar D, C, B, A, iar dintre ieșirile Y1—Y8, doar Y1—Y7, conectate respectiv la intrările a, b, c, d, e, f, g ale circuitului de afișare cu 4 segmente. Înainte de înscriere, memoriile conțin valoarea 0 în toate locațiile.

Conectarea liniilor de adresă A0—A3, A4—A7, A8—A11, A12—A15 sau de date D0—D3, D4—D7 la circuitul format din memoria PROM și circuitul de afișare, este reprezentată în figura 8.36. Sistemul va conține 6 astfel de circuite. Afișarea unor semnale din sistem ca  $\overline{\text{MREQ}}$ ,  $\overline{\text{IORQ}}$ , RD, WR sau altele, este ilustrată în aceeași figură.

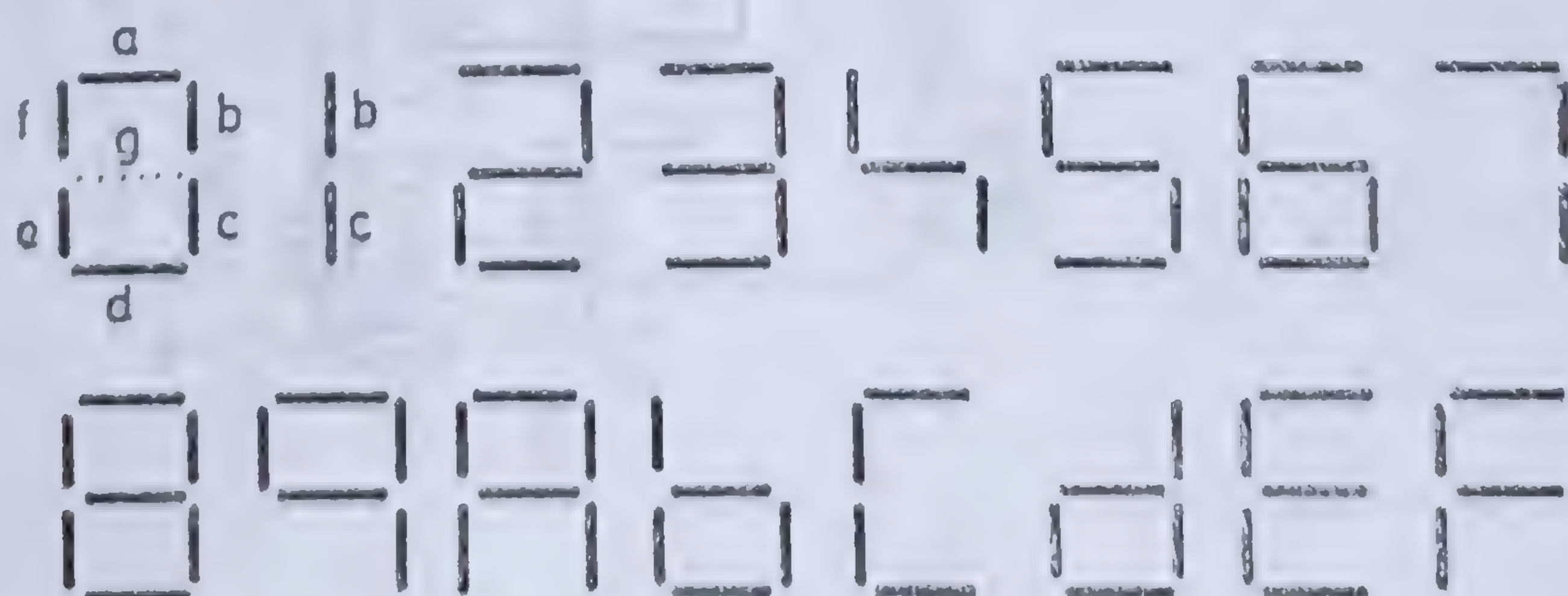


Fig. 8.34 Formatul cifrelor hexazecimale pentru afișare pe 7 segmente.



E	D	C	B	A	y <sub>1</sub> a	y <sub>2</sub> b	y <sub>3</sub> c	y <sub>4</sub> d	y <sub>5</sub> e	y <sub>6</sub> f	y <sub>7</sub> g	y <sub>8</sub>
0	0	0	0	0	0	0	0	0	0	0	1	x
0	0	0	0	1	1	0	0	1	1	1	1	x
0	0	0	1	0	0	0	1	0	0	1	0	x
0	0	0	1	1	0	0	0	0	1	1	0	x
0	0	1	0	0	1	1	0	1	1	0	0	x
0	0	1	0	1	0	1	0	0	1	0	0	x
0	0	1	1	0	0	1	0	0	0	0	0	x
0	0	1	1	1	0	0	0	1	1	1	1	x
0	1	0	0	0	0	0	0	0	0	0	0	x
0	1	0	0	1	0	0	0	1	1	0	0	x
0	1	0	1	0	0	0	0	1	0	0	0	x
0	1	0	1	1	1	1	0	0	0	0	0	x
0	1	1	0	0	0	1	1	0	0	0	1	x
0	1	1	0	1	1	0	0	0	0	1	0	x
0	1	1	1	0	0	1	1	0	0	0	0	x
0	1	1	1	1	0	1	1	1	0	0	0	x

Fig. 8.35 Conținutul memoriilor PROM 74188.

### Funcționarea pas cu pas a microprocesorului Z80, instrucțiune cu instrucțiune realizată hardware, cu refreșarea memoriilor RAM dinamic

Schema prezentată în continuare, are față de cea anterioară, avantajul refreșării memoriilor RAM dinamic, deci posibilitatea de a rula „pas cu pas” un program înscris în astfel de memorii. Funcționarea pas cu pas se realizează prin executarea unei instrucțiuni de salt relativ, în mod repetat, de un număr nedefinit de ori, la adresa de început a instrucțiunii pe care se dorește oprirea programului. Inserarea instrucțiunii de salt relativ „JR -2” se realizează hard, și este posibilă numai după executarea completă a unei instrucțiuni din memoria sistemului. Ca urmare, la fiecare „pas” se va executa o instrucțiune în întregime, spre deosebire de cazul prezentat anterior, în care la fiecare „pas” se execută câte un ciclu de mașină.

Schema propusă este prezentată în figura 8.37.

Schema se poate implementa în orice sistem în care magistrala de date este prevăzută cu registre tampon bidirecționale, cu ieșiri cu 3 stări. În schema prezentată, acestea sînt circuitele 8216, care separă unitatea centrală de memorii (și eventual de dispozitivele de I/E). Datele traversează aceste circuite dacă semnalul  $\overline{PP}=0$  (pas cu pas) deci microprocesorul nu funcționează pas cu pas și dacă  $\overline{MREQ}=0$  sau  $\overline{IORQ}=0$  (acest ultim semnal nu este necesar dacă circuitele 8216 separă numai memoriile de microprocesor). Sensul de circulație al datelor este dat de semnalul

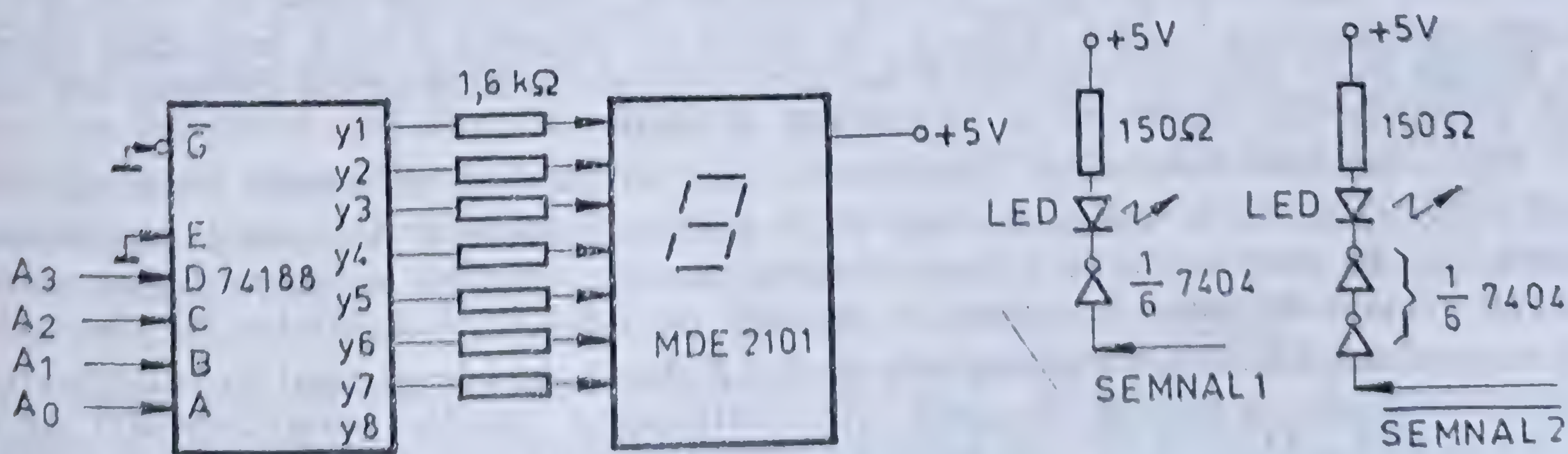


Fig. 8.36 Afișarea adreselor, datelor și a unor semnale de control din sistem.







frontul crescător aplicat pe intrarea de tact face ca valoarea 0 aplicată pe intrarea D să se înscrie în bistabil, deci  $P/\bar{A}=0$ ; pe poziția „AUTOMAT” a comutatorului „P/A”, se mai înscrie bistabilul, deci  $P/\bar{A}=1$ , dacă se închide comutatorul „STOP PE ADRESA” și semnalul  $\overline{EQU}$ , cu aceeași semnificație și proveniență ca în figura 8.31, ia valoarea 0 (atunci când apare coincidența între adresa din sistem și o adresă de stop fixată cu 16 comutatoare). Circuitul trebuie să conțină și partea de generare a semnalului de coincidență a adreselor. Cu comutatorul „STOP PE ADRESA” deschis, semnalul  $\overline{EQU}$  nu afectează bistabilul B1. Dacă la aplicarea semnalului  $\overline{RESET}$ , comutatorul P/A este pe poziția „PAS CU PAS”, atunci  $P/\bar{A}=1$  atât înainte, în timpul, cât și după aplicarea semnalului  $\overline{RESET}$ .

După oprirea pe o adresă fixată anterior,  $P/\bar{A}=1$ , Fixînd o nouă adresă de stop, trecerea comutatorului „P/A” pe poziția „PAS CU PAS” și revenirea pe poziția „AUTOMAT” șterge bistabilul B1, determinînd rularea programului din memorie, automat, pînă la atingerea noii adrese de stop, când  $P/\bar{A}=1$  și se trece în regim pas cu pas.

Dacă semnalul  $P/\bar{A}=0$ , bistabilul B2 este înscris; primul front căzător al lui  $\overline{M1}$  face să se înscrie și B3, deci  $\overline{PP}=0$ , permițînd circulația datelor prin circuitele 8216, iar  $\overline{PP}=1$ , determinînd ieșirile circuitului 8212 să se afle în starea de impedanță ridicată. Microprocesorul rulează în acest caz programul din memorie. De notat că  $\overline{PP}=0$  se aplică pe intrarea  $\overline{CLR}$  al lui B2 simultan cu  $P/\bar{A}=0$  pe  $\overline{PR}$ , ceea ce menține ieșirea Q a lui B2 la 1 logic. Intrarea CK a lui este inactivă.

Dacă semnalul  $P/\bar{A}=1$ ,  $\overline{PP}=0$  aplicat anterior pe intrarea  $\overline{CLR}$  a lui B1 îl șterge, și primul front căzător al lui  $\overline{M1}$  (deci crescător al lui  $\overline{M1}$ ) înscrie bistabilul B3, ceea ce face  $\overline{PP}=0$  și  $\overline{PP}=1$ . Semnalul  $\overline{PP}=1$  blochează circuitele 8216, deci microprocesorul nu mai poate prelua instrucțiuni din memorie, iar  $\overline{PP}=0$  deschide circuitul 8212. Citirea codului operației de către unitatea centrală se face deci din circuitul 8212 și este  $00011000_B(18_H)$ , pentru că  $\overline{M1}=0$ . În timpul primului ciclu de mașină al instrucțiunii cu codul operației  $18_H(JR, \text{ salt relativ})$ , are loc și reîmproșarea memoriilor dinamice. Următorul octet este interpretat ca deplasament ( $e=-2$ ) pentru această instrucțiune și este citit cu  $\overline{M1}=1$ , tot din circuitul 8212, deci va fi  $11111110_B(FE_H)$ , ceea ce reprezintă în complement față de 2, valoarea  $-2$ . Ca urmare se va face un salt la adresa instrucțiunii care urma ultimei instrucțiuni executate din memoria sistemului.

O apăsare pe butonul fără reținere „1 PAS”, prevăzut cu un bistabil RS format din 2 porți SI NU, pentru deparazitare, determină aplicarea unui front crescător pe intrarea de tact a bistabilului B2, deci înscrierea lui. Valoarea 1 de la ieșirea lui B2 se înscrie în B3 pe primul front căzător al lui  $\overline{M1}$ , deschizînd circuitele 8216 și închizînd circuitul 8212. În același timp,  $\overline{PP}=0$  șterge bistabilul B2, aplicînd 0 pe intrarea D a lui B3. Înscrierea acestuia are loc doar la următorul front căzător al lui  $\overline{M1}$ , deci după executarea completă a unei instrucțiuni din memoria sistemului. La începutul următorului ciclu de instrucțiune,  $\overline{M1}=0$ , frontul căzător al lui  $\overline{M1}$  șterge și bistabilul B3, blocînd circuitele 8216 și deschizînd circuitul 8212, reîncepînd astfel execuția instrucțiunii de salt relativ. În timpul executării acestei instrucțiuni, se poate realiza afișarea adreselor și datelor din sistem (datele preluate din memoria sistemului) atunci când  $\overline{M1}=0$ . Se afișează astfel doar codul operației la fiecare pas. Un circuit mai complex ar putea afișa, după memorare, și ceilalți octeți ai fiecărei instrucțiuni. Circuitul pentru afișarea codului operației este cel din figura 8.38. Circuitul este asemănător cu cel din figura 8.36, cu deosebirea că afișarea are loc numai când  $\overline{M1}=0$ , deci pentru primul octet al fiecărei instrucțiuni.



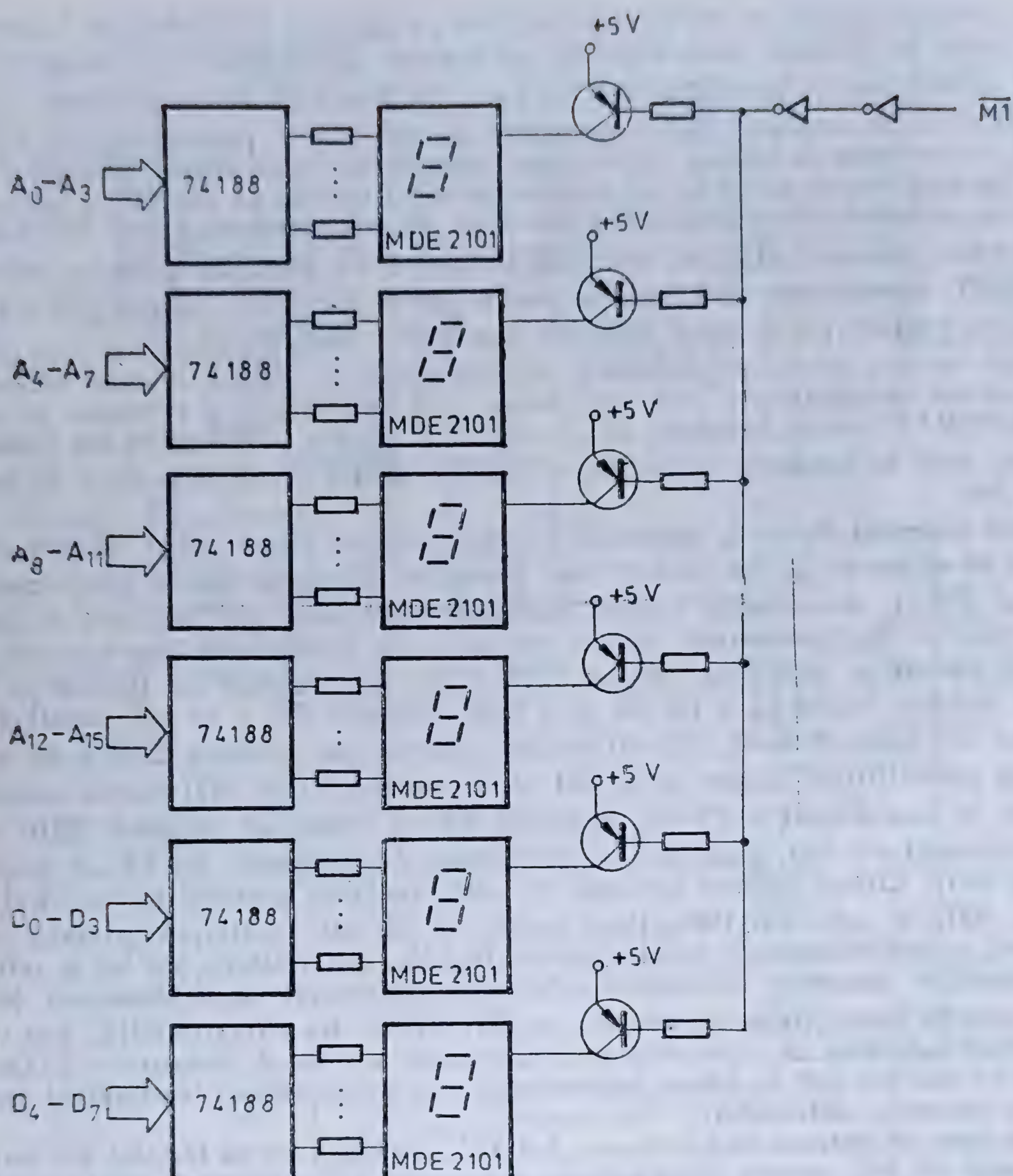


Fig. 8.38. Afișarea datelor și adreselor în regim „cite o instrucțiune”.

### Comanda PWM a unui servomotor de curent continuu

● metoda de comandă a motoarelor de curent continuu, adaptabile sistemelor de control numerice, este metoda PWM (pulse width modulation) — cu modularea lățimii impulsurilor. Dintre variantele existente, se prezintă un exemplu de comandă cu punte în H, care permite, prin aplicarea unui semnal de comandă pe una sau pe alta din diagonalele punții, alimentarea motorului cu tensiune de polaritatea pozitivă sau negativă. Prin alternarea semnalelor de comandă ale diagonalelor, cu o frecvență care se determină prin calcul (uzual de ordinul zecilor de kHz), și prin modificarea factorului de umplere  $T_+/T$  se obține pe motor o tensiune în gama  $-U \div +U$ . Un amplificator în H utilizabil pentru alimentarea motorului și forma semnalelor de comandă ale punții înt prezentate în figura 8.39.



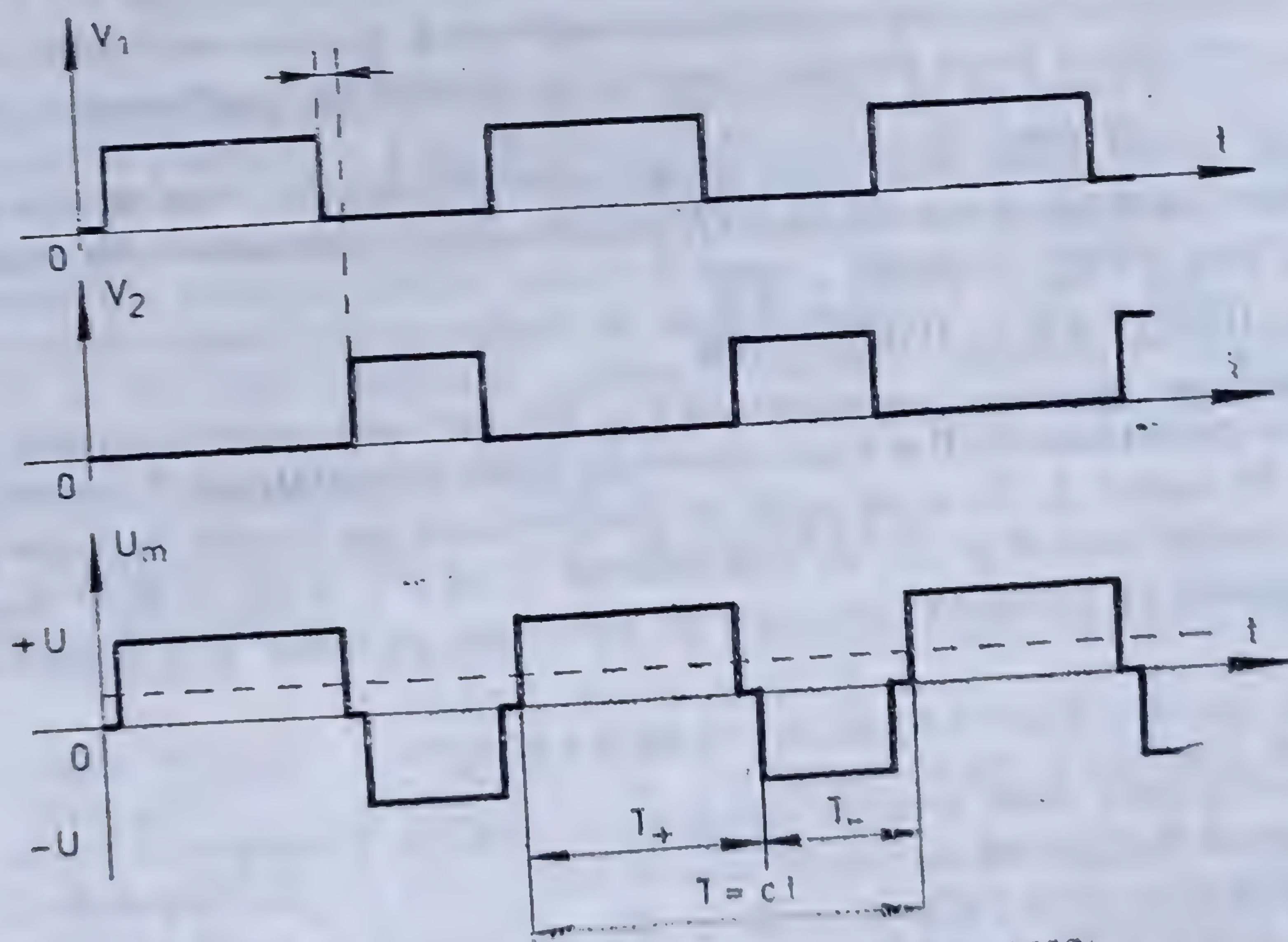
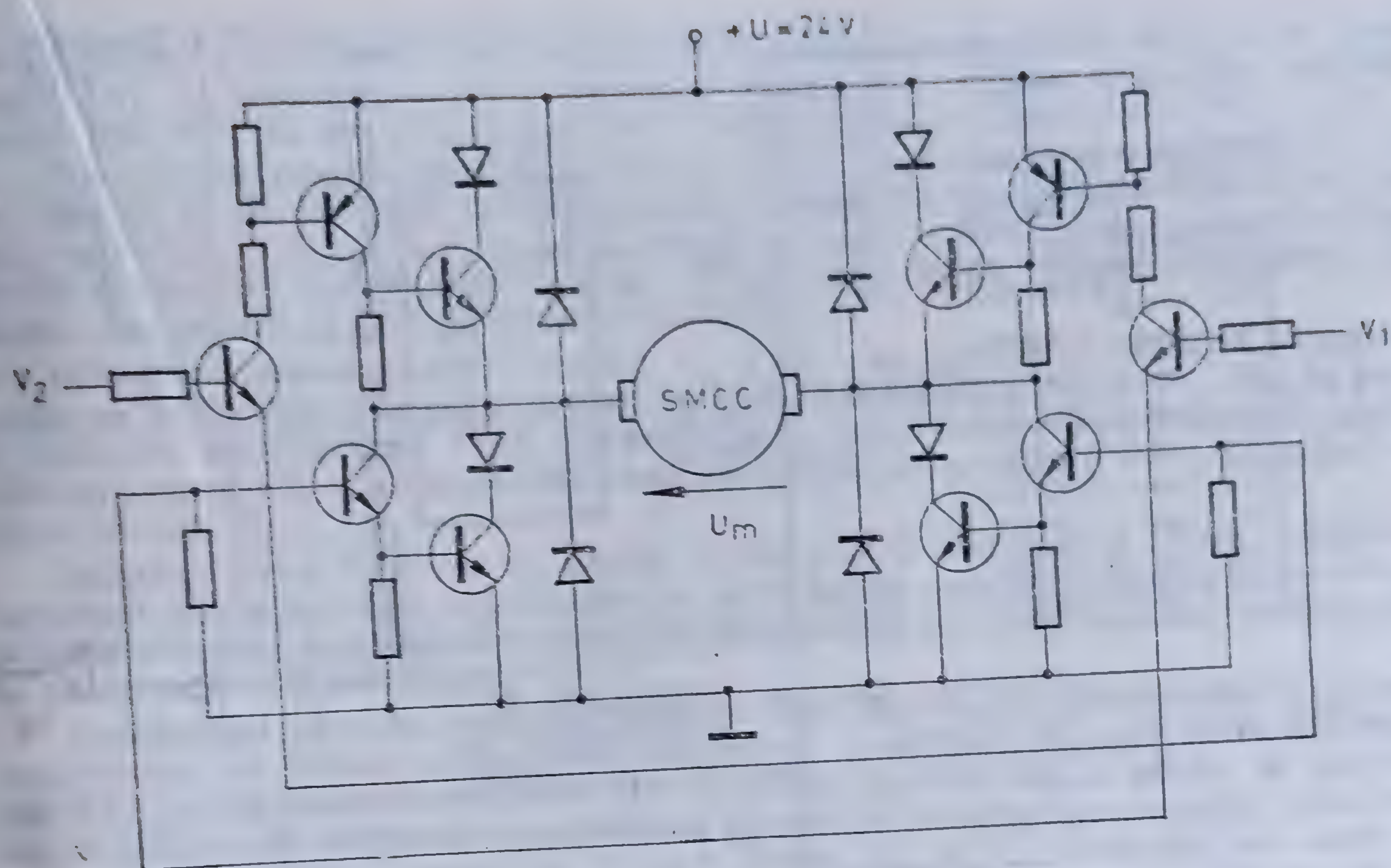


Fig. 8.39 Punte în H pentru comanda PWM a MCC.

Tensiunea medie pe motor este

$$U_d = U \left( 2 \cdot \frac{T_+}{T} - 1 \right)$$

pentru  $T_+ = 0$  se obține  $U_d = -U$ , iar pentru  $T_+ = T$ ,  $U_d = +U$ . Raportul  $T_+/T$  poate fi controlat de sistemul cu microprocesor care comandă motorul. Schema propusă



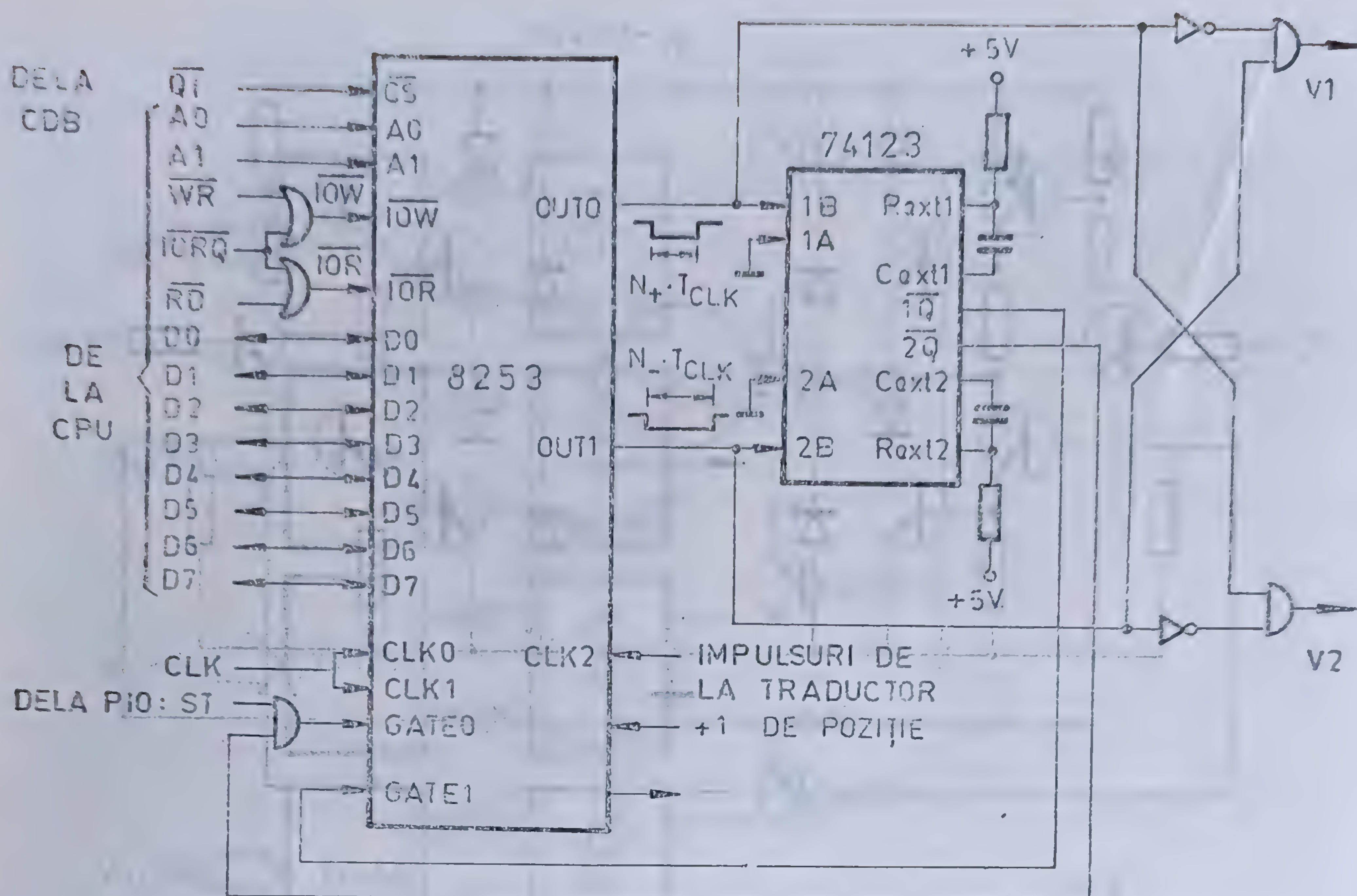


Fig. 8.40 Circuitul pentru generarea semnalelor de comandă ale amplificatorului PWM.

utilizează un circuit 8253, numărător programabil din familia microprocesorului 8080. Se poate observa generarea semnalelor IOW și IOR necesare componentelor familiei 8080:

$$\begin{aligned} \overline{\text{IOR}} &= \overline{\text{IORQ}} \cdot \overline{\text{RD}} = \overline{\text{IORQ}} + \overline{\text{RD}} \\ \overline{\text{IOW}} &= \overline{\text{IORQ}} \cdot \overline{\text{WR}} = \overline{\text{IORQ}} + \overline{\text{WR}} \end{aligned}$$

Circuitul care generează semnalele V1 și V2 este prezentat în figura 8.40. Sînt utilizate doar numărătoarele 0 și 1 ale circuitului 8253 (numărătorul 2 nu este utilizat), programate în modul 1. În acest mod, se programează un număr în fiecare numărător ( $N_+$  în numărătorul 0 și  $N_-$  în numărătorul 1, cu  $N_+ + N_- = N = \text{ct}$ ). Suma  $N$  a lor multiplicată cu perioada frecvenței de tact, este perioada  $T$  a semnalelor V1 și V2:

$$T = (N_+ + N_-) \cdot T_{\text{CLK}}$$

La aplicarea unui front crescător pe intrarea GATE, numărătorul începe să funcționeze și asigură la ieșire un semnal 0 logic de durată  $N_+ \cdot T_{\text{CLK}}$ , respectiv  $N_- \cdot T_{\text{CLK}}$ . Monostabilele din circuitul 74123 au rolul să asigure o pauză de ordinul microsecundelor între semnalele V1 și V2, durata ei fiind:

$$\Delta t = 0,32 \cdot R \cdot C \left( 1 + \frac{0,7}{R} \right), \text{ cu } [R] = 1 \text{ K}\Omega, [C] = 1 \text{ pF}, [t] = 1 \text{ ns}$$

Acest interval de timp între perioadele active ale lui V1 și V2 este necesar pentru protejarea amplificatorului PWM, deoarece deschiderea simultană a diagonalelor lui ar cauza avaria amplificatorului. Rolul porților SI de la ieșirea circuitului este de a împiedica apariția simultană a valorilor  $V1 = V2 = 1$ .



Declanșarea funcționării motorului și oprirea lui se poate realiza cu ajutorul unui semnal, ST, de start/stop, cu stare de repaus 1 logic și care poate fi o ieșire dintr-un circuit Z80 PIO, cu un port programat ca intrare/ieșire de bit.

Presupunând programate număratoarele 0 și 1 în modul 1 și constantele  $N_+$  și  $N_-$  înscrise în ele, un front crescător pe ST declanșează numărătorul 0 și  $V1=1$  un timp egal cu  $N_+ \cdot T_{CLK}$ . La terminarea acestei perioade, frontul crescător al semnalului OUT0 declanșează monostabilul 1, care, cu întârzierea  $\Delta t$ , declanșează numărătorul 1. Acesta declanșează la fel numărătorul 0, ș.a.m.d.

Schimbarea din mers a constantelor  $N_+$  și  $N_-$  este posibilă și modifică tensiunea medie pe motor, deci viteza lui. Oprirea se poate face fie dacă  $N_+ = N_-$ , caz în care  $U_a = 0$ , dar amplificatorul PWM funcționează, fie prin blocarea număratoarelor, aplicînd un semnal  $ST = 0$ , cu o durată suficientă ca să nu permită o nouă declanșare a numărătorului 0.

Includerea unui traductor de poziție în sistemul de comandă și citirea „poziției” motorului prin numărarea impulsurilor de la traductor permite realizarea unui sistem de poziționare pe o axă. Dublarea circuitului de comandă permite realizarea unui sistem de poziționare în 2 coordonate.

Presupunînd adresele circuitului 8253 ca fiind 04, 05, 06, 07<sub>H</sub> (primele 3 pentru număratoare, iar ultima pentru portul de control) și utilizînd portul A al unui circuit Z80 PIO pentru semnalul ST (bitul A0), cu adresa de date C0<sub>H</sub> și adresa de control C1<sub>H</sub>, se prezintă în continuare programarea circuitelor în cadrul programului principal, subrutina de pornire a motorului ( $U_a = 0$  pentru  $N_+ = N_-$ ), subrutina de mers cu viteză maximă în sensul ales pozitiv ( $U_a = +U$ ) și cea de mers în sens negativ cu viteză redusă la jumătate ( $U_a = -\frac{1}{2}U$ ). Subrutine asemănătoare pot completa gama de viteze pe care o poate atinge motorul, în funcție de informațiile de la traductorul de poziție, în timpul ciclului de poziționare. Impulsurile de la traductor pot fi numărate, de exemplu, cu numărătorul 2 al circuitului 8253, ceea ce oferă posibilitatea citirii din mers a poziției. Înainte de poziționare, numărătorul 2 programat în modul 0 este încărcat cu numărul maxim (de exemplu 65535 pentru 2 octeți), diferența dintre acest număr și numărul citit fiind numărul de impulsuri de la traductor.

Suma  $N_+ + N_-$  este constantă, pentru a asigura perioada  $T$  constantă. Cu  $N_+ + N_- = 200$ , se obține  $T = 200 \cdot T_{CLK}$ , deci frecvența de comandă PWM va fi  $f = f_{CLK}/200 = 2500000/200 = 12500 \text{ Hz}$ .

#### ETICHETA COD OPERATIE COMENTARIU PROGRAMUL PRINCIPAL

```

PROGPR: LD A,32H      ; CUVINT DE PROGRAMARE PENTRU NUMARA-
                  ; TORUL 0 IN MODUL 1
          OUT (07H),A  ; PROGRAMEAZA NUMARATORUL 0 IN MODUL 1
          LD A,72H     ; CUVINT DE PROGRAMARE PENTRU NUMARA-
                  ; TORUL 1 IN MODUL 1
          OUT (07H),A  ; PROGRAMEAZA NUMARATORUL 1 IN MODUL 1
          LD A,CFH     ; CUVINT DE CONTROL IN MOD 3 PENTRU PIO
          OUT (C1H),A  ; PROGRAMEAZA CANALUL A DIN PIO IN MODUL 3
          LD A,00H     ; CUVINT DE CONTROL AL REGISTRULUI I/E
          OUT (C1H),A  ; PROGRAMEAZA BITII CANALULUI A CA IESIRI
          LD A,03H     ; CUVINT DE DEZACTIVARE A INTRERUPERILOR
          OUT (C1H),A  ; DEZACTIVEAZA INTRERUPERILE LA CANALUL A
          LD A,FFH     ; INSCRIE FFH IN ACUMULATOR
          OUT (C0H),A  ; ADUCE LA 1 IESIRILE CANALULUI A DIN PIO
          LD A,64H     ; OCTET INFERIOR DIN  $N_+ = N_- = 100$  IN A
          OUT (04H),A  ; PROGRAMEAZA NUMARATORUL 0 CU OCTETUL
                  ; INF.

```



OUT (05H),A ; PROGRAMEAZA NUMARATORUL 1 CU OCTETUL  
 ; INF.  
 LD A,00H ; OCTET SUPERIOR DIN  $N+ = N- = 100$  IN ACUMU-  
 ; LATOR  
 OUT (04H),A ; PROGRAMEAZA NUMARATORUL 0 CU OCTETUL  
 ; SUP.  
 OUT (05H),A ; PROGRAMEAZA NUMARATORUL 1 CU OCTETUL  
 ; SUP.

CALL START ; APELEAZA SUBROUTINA DE GENERARE A SEM-  
 ; NALULUI DE START

CALL VMXP ; APELEAZA SUBROUTINA DE COMANDA A MCC  
 ; CU VITEZA MAXIMA IN SENS POZITIV

CALL VRJN ; APELEAZA SUBROUTINA DE COMANDA A MCC CU  
 ; VITEZA REDUSA LA JUMATATE, SENS NEGATIV

CALL STOP ; APELEAZA SUBROUTINA DE GENERARE A SEM-  
 ; NALULUI DE BLOCARE

#### SUBROUTINA DE START

START: LD A,FEH ; FACE  $A0 = 0$ , RESTUL BITILOR SINT 1  
 OUT (C0H),A ; OPERATIE DE IESIRE, FACE  $ST = 0$   
 LD A,FFH ; REFACE  $A0 = 1$   
 OUT (C0H),A ; FACE  $ST = 1$ , FRONTUL CRESCATOR DECLANSEA-  
 ; ZA NUMARATORUL 0  
 RET ; REVINE IN PROGRAMUL PRINCIPAL

#### SUBROUTINA DE COMANDA A MCC CU VITEZA MAXIMA IN SENS POZITIV

VMXP: LD A,C7H ; OCTETUL INFERIOR DIN  $N+ = 199$ , IN A  
 OUT (04H),A ; PROGRAMEAZA NUMARATORUL 0 CU OCTETUL  
 ; INF.  
 LD A,00H ; OCTETUL SUPERIOR DIN  $N+ = 199$ , IN A  
 OUT (04H),A ; PROGRAMEAZA NUMARATORUL 0 CU OCTETUL  
 ; SUP.  
 LD A,01H ; OCTETUL INFERIOR DIN  $N- = 1$ , IN A  
 OUT (05H),A ; PROGRAMEAZA NUMARATORUL 1 CU OCTETUL  
 ; INF.  
 LD A,00H ; OCTETUL SUPERIOR DIN  $N- = 1$  IN A  
 OUT (05H),A ; PROGRAMEAZA NUMARATORUL 1 CU OCTETUL  
 ; SUP.



RET ; REVINE IN PROGRAMUL PRINCIPAL

SUBROUTINA DE COMANDA A MCC CU VITEZA REDUSA LA JUMATATE,  
IN SENS NEGATIV

VRN: LD A,32H ; OCTETUL INFERIOR DIN  $N+=50$ , IN A  
OUT (04H),A ; PROGRAMEAZA NUMARATORUL 0 CU OCTETUL  
; INF.  
LD A,00H ; OCTETUL SUPERIOR DIN  $N+=50$ , IN A  
OUT (04H),A ; PROGRAMEAZA NUMARATORUL 0 CU OCTETUL  
; SUP.  
LD A,96H ; OCTETUL INFERIOR DIN  $N-=150$ , IN A  
OUT (05H),A ; PROGRAMEAZA NUMARATORUL 1 CU OCTETUL  
; INF.  
LD A,00H ; OCTETUL SUPERIOR DIN  $N-=150$ , IN A  
OUT (05H),A ; PROGRAMEAZA NUMARATORUL 1 CU OCTETUL  
; SUP.  
RET ; REVINE IN PROGRAMUL PRINCIPAL

SUBROUTINA DE OPRIRE

STOP: LD A,FEH ; FACE A0=0, RESTUL BITILOR EGALI CU 1  
OUT (C0H),A ; FACE ST=0 PENTRU OPRIREA NUMARATORU-  
; LUI 0  
CALL TEMP ; APELEAZA O SUBROUTINA DE TEMPORIZARE  
; PENTRU A MENTINE ST=0 O DURATA MAI MARE  
; DECIT DE N ORI TCLK  
RET ; REVINE IN PROGRAMUL PRINCIPAL

### Realizarea unui interpolator liniar pentru comanda motoarelor pas cu pas

Interpolarea liniară este utilizată în echipamentele de poziționare în 2 sau 3 coordonate. Schema prezentată este utilizabilă pentru interpolare în 2 coordonate, acționarea fiind realizată cu motoare pas cu pas.

Principiul utilizat este următorul: presupunând necesară descrierea unui segment ale cărui proiecții pe axe sînt NX și NY (în număr de pași ai MPP), se notează  $NMAX = \max(NX, NY)$  și  $NMIN = \min(NX, NY)$ . Se generează un tren de NMAX impulsuri, cu frecvența maximă admisă de motoare și se aplică pe axa cu mai mulți pași. Pe axa cu NMIN pași se aplică impulsul numai cînd există o „depășire” în suma

$$\Sigma NMIN \pmod{NMAX}$$

deci cînd trebuie reajustată suma pentru a o aduce în domeniul  $[0, NMAX-1]$ . Calculul acestei sume începe de la valoarea  $[NMAX/2]$  (parte întreagă).

Un exemplu pentru  $NX=8$  și  $NY=3$  este prezentat în figura 8.41. Curba descrisă în plan și care aproximează segmentul dorit este reprezentată în figura 8.42.

Schema interpolatorului este prezentată în figura 8.43. Se utilizează parțial un numărator 8253: numărătorul 0, programat în modul 3, divizează frecvența generatorului de tact, CLK, prin  $[CLK/f_{max}]$ , unde  $f_{max}$  este frecvența maximă care se aplică distribuitorului de impulsuri al motoarelor pas cu pas. Se obține astfel  $f_{max}^* \approx f_{max}$ . Numărătorul 1 este programat în modul 1, cu constanta  $NMAX$  generînd la ieșirea OUT un impuls 0 logic de durată  $NMAX \cdot T$ , unde  $T = 1/f_{max}^*$ . O poartă SI decupează din trenul de impulsuri  $f_{max}^*$  un număr de NMAX impulsuri, care se aplică unuia dintre



$\Sigma NMIN(\text{mod } NMAX)$	DEPĂȘIRE?	NR. PAS PE AXAX	NR. PAS PE AXAY
$4 + 3 (\text{mod } 8) = 7$	NU	1	
$7 + 3 (\text{mod } 8) = 2$	DA	2	1
$2 + 3 (\text{mod } 8) = 5$	NU	3	
$5 + 3 (\text{mod } 8) = 0$	DA	4	2
$0 + 3 (\text{mod } 8) = 3$	NU	5	
$3 + 3 (\text{mod } 8) = 6$	NU	6	
$6 + 3 (\text{mod } 8) = 1$	DA	7	3
$1 + 3 (\text{mod } 8) = 4$	NU	8	

Fig. 8.41 Exemplu de aplicare a algoritmului de interpolare, pentru  $NX=8$ ,  $NY=3$ .

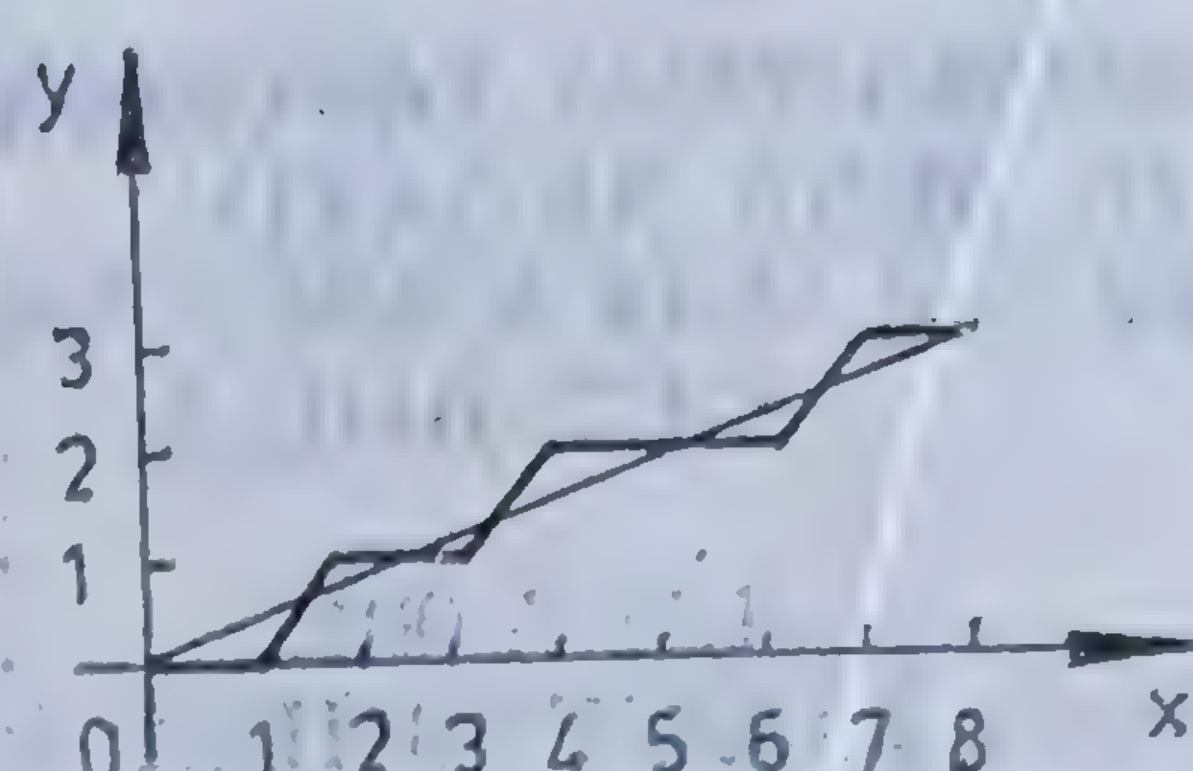


Fig. 8.42 Aproximarea unui segment de dreaptă.

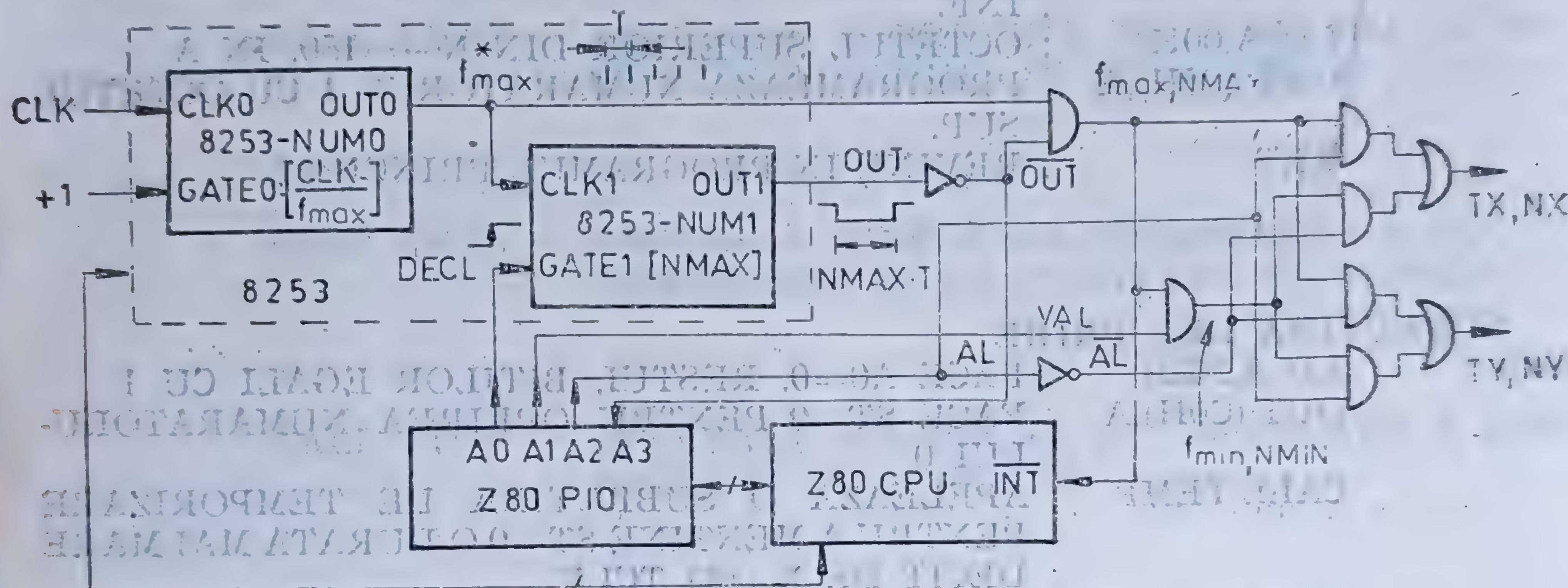


Fig. 8.43 Interpolator liniar pentru comanda motoarelor pas cu pas.

motoarele pas cu pas. Terminarea fiecărui pas pe această axă generează o întrerupere prin aplicarea unui 0 logic pe intrarea INT a microprocesorului. Acesta calculează  $\Sigma NMIN (\text{mod } NMAX)$  și determină dacă următorul pas se execută și pe axa cu NMIN pași, prin valoarea semnalului VAL (validare). Semnalul VAL este generat de un port al unui circuit Z80 PIO, programat în modul 3 (intrare/ieșire de bit). Alte 2 ieșiri ale acestui port sînt DECL (declanșare), pentru declanșarea număratorului 1 pe frontul pozitiv aplicat pe intrarea GATE1 și AL (alegere) care permite aplicarea frecvenței maxime pe axa X sau pe axa Y. Un bit al aceluiași port poate fi utilizat ca intrare pentru semnalul OUT, a cărui valoare 1 indică existența unei mișcări, fapt ce trebuie cunoscut pentru a nu declanșa o mișcare înainte de terminarea celei anterioare. În varianta de mai sus, validarea întreruperilor printr-o instrucțiune EI trebuie efectuată înaintea executării primului pas.

Pornirea și oprirea cu frecvență variabilă a motoarelor pas cu pas este posibilă prin modificarea, din mers, a constantei cu care NUM0 divizează frecvența CLK: scăzînd valoarea ei, frecvența  $f_{max}^*$  crește (pornire MPP) și crescînd-o,  $f_{max}^*$  scade (oprire). Momentele în care trebuie schimbată constanta de divizare se pot determina comparînd numărul de pași efectuați sau cel pași care trebuie efectuați (acesta din urmă se citește direct din număratorului NUM1, în timpul funcționării lui), cu constante impuse de profilul de viteză ales.



[illegible]

Frecvența aplicată distribuitorului de impulsuri poate fi diferită pentru cele 6 motoare, divizînd frecvența CLK cu diferite constante. Numărul de pași poate fi de asemenea diferit pe cele 6 axe. Frecvența de comandă poate fi modificată din mers, schimbînd constanta de divizare la primul numărător (programat în modul 3) al fiecărei perechi de pe o axă. Pe fiecare axă se poate executa un număr de pași în domeniul  $[1,65536]$ , cu o frecvență minimă de  $2500000/65536 \approx 38$  impulsuri/secundă.

Semnalele „OUT” pot fi utilizate, ca și în schema de interpolare, pentru citire cu ajutorul unui port Z80 PIO, pentru a determina dacă există mișcare pe oricare dintre axe.

Declanșarea independentă a numărătoarelor programate în modul 1 (al doilea din fiecare pereche) permite efectuarea de mișcări separate sau simultane pe axe, fiind posibilă și corelarea lor, prin program.

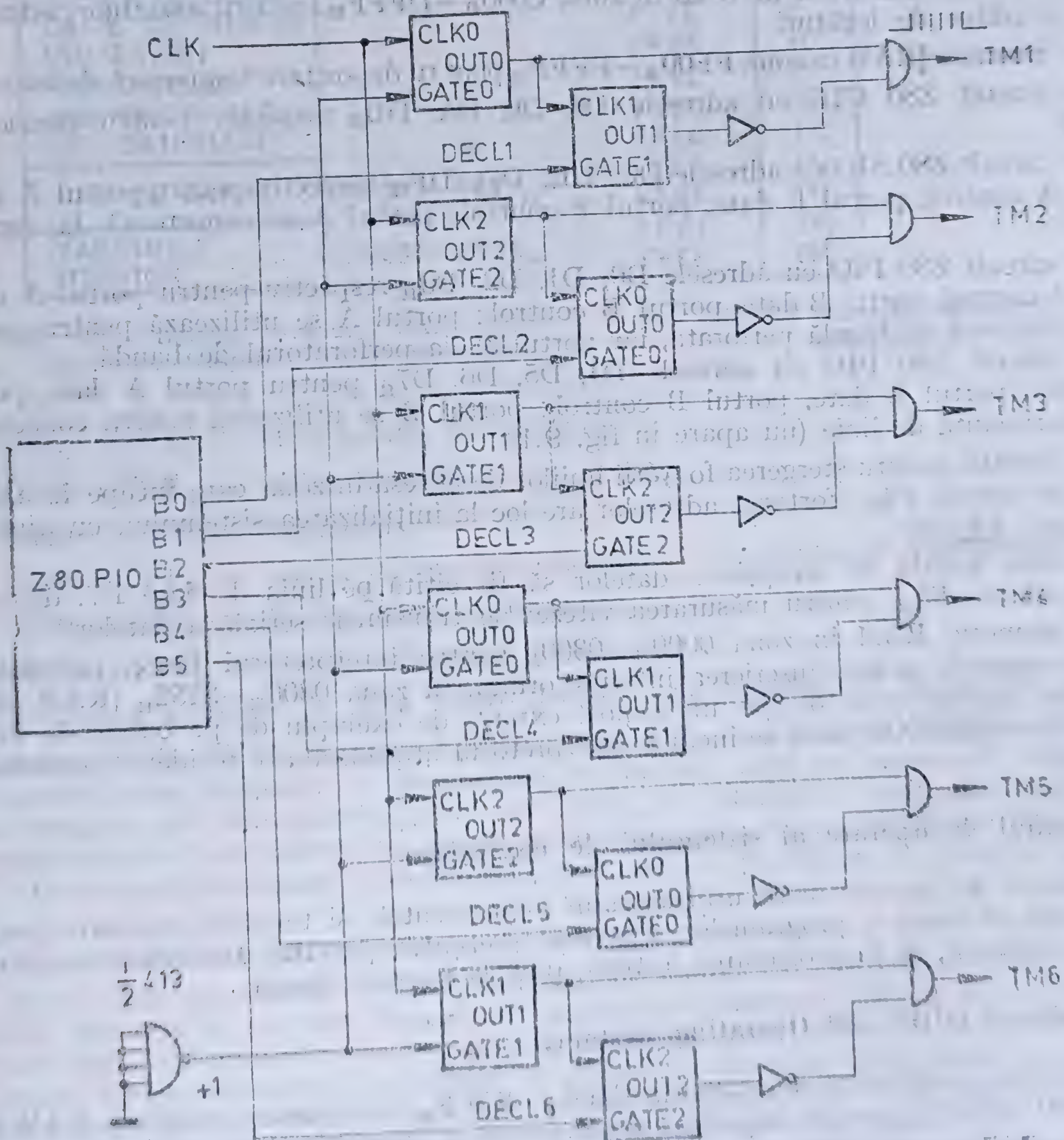


Fig. 8.44 Schemă de comandă pentru un sistem de acționare cu 6 motoare pas cu pas.



## CAPITOLUL IX

### SISTEM DE DEZVOLTARE CU MICROPROCESOR Z80

#### 9.1. STRUCTURA HARDWARE ȘI SISTEMUL DE OPERARE

##### Structura hardware a sistemului de dezvoltare cu microprocesor Z80

Schema unui sistem bazat pe microprocesorul Z80 este prezentată în figura 9.1. Sistemul este conceput astfel încât să poată fi utilizate un program monitor, un editor de texte, un asamblor, un editor de legături și un interpretor Basic.

Pentru ca implementarea acestor programe să fie posibilă, trebuie respectate următoarele condiții în structura hardware:

- memorie RAM de la  $0000_H$  (în schemă, de la  $0000_H$  la  $BFFF_H$ , deci 48 ko)
- memorie EPROM de 2 ko în zona  $E000_H$ — $E7FF_H$ , pentru monitor
- memorie EPROM de 8 ko în zona  $C000_H$ — $DFFF_H$ , pentru asamblor, editor de texte și editor de legături
- memorie RAM în zona  $FF00_H$ — $FFFF_H$  pentru depozitare temporară de informații
- circuit Z80 CTC cu adresele D8, D9, DA, DB<sub>H</sub> respectiv pentru canalele 0, 1, 2, 3
- circuit Z80 SIO cu adresele DC, DE, DD, DF<sub>H</sub> respectiv pentru portul A date, portul A control, portul B date, portul B control; portul A se conectează la consola serială
- circuit Z80 PIO cu adresele D0, D1, D2, D3<sub>H</sub> respectiv pentru portul A date, portul A control, portul B date, portul B control; portul A se utilizează pentru conectare la cititorul de bandă perforată, iar portul B, la perforatorul de bandă
- circuit Z80 PIO cu adresele D4, D5, D6, D7<sub>H</sub> pentru portul A date, portul A control, portul B date, portul B control; portul B se utilizează pentru conectarea la o imprimantă de linie (nu apare în fig. 9.1)
- bistabil pentru ștergerea forțării liniilor de adresă în zona care începe la adresa  $E000_H$ , cu adresa  $F4_H$ ; forțarea adreselor are loc la inițializarea sistemului, cu ajutorul semnalului RESET
- linia serială de recepție a datelor să fie citită pe linia de date D7, dintr-un port cu adresa  $F5_H$ , pentru măsurarea vitezei de transmisie serială a datelor
- memorie RAM în zona  $0000_H$ — $0300_H$  pentru interpretorul Basic, ca zonă de lucru și memorie pentru înscrierea interpretorului, în zona  $0300_H$ — $3152_H$  (RAM, dacă se citește interpretorul de pe un suport extern, de exemplu de pe bandă de hîrtie perforată, sau EPROM dacă se include interpretorul în programele rezidente permanente în sistem).

##### Sistemul de operare al sistemului de dezvoltare

Sistemul de operare oferă utilizatorului instrumentele și tehnicile necesare pentru dezvoltarea eficientă a programelor. Sistemul prezentat permite utilizatorului dezvoltarea de aplicații, de la proiectarea inițială, pînă la testarea finală.

##### Monitorul (DDT—80 Operating System)

Monitorul ocupă zona de memorie  $E000_H$ — $E7FF_H$  și utilizează memoria RAM din zona  $FF00$ — $FFFF_H$  pentru depozitarea temporară a unor informații, inclusiv a conținutului registrelor microprocesorului. Utilizarea memoriei este următoarea:



CONTINUTUL REGISTRELOR UTILIZATE	
STIVA MONITORULUI	
STIVA UTILIZATOR	
(LIMITA DE DEMAR- CATIE DEFINITA DE UTILIZATOR)	
TABELELE DE MNEMONICE ALE UTILIZATORULUI	
I/E DE CANAL	
VARIABLE MONITOR	

FFE6 - FFFF<sub>H</sub>

FF90 - FFE5<sub>H</sub>

FF33 - FF8F<sub>H</sub>

FF27 - FF32<sub>H</sub>

FF00 - FF26<sub>H</sub>

FFFF  
FFFE  
FFFD  
FFFC  
FFFB  
FFFA  
FFF9  
FFF8  
FFF7  
FFF6  
FFF5  
FFF4  
FFF3  
FFF2  
FFF1  
FFF0  
FFEF  
FFEE  
FFED  
FFEC  
FFEB  
FFEA  
FFE9  
FFE8  
FFE7  
FFE6

PC  
PC  
A  
F  
I  
IF  
B  
C  
D  
E  
H  
L  
A'  
F'  
B'  
C'  
D'  
E'  
H'  
L'  
IX<sub>H</sub>  
IX<sub>L</sub>  
IY<sub>H</sub>  
IY<sub>L</sub>  
SP<sub>H</sub>  
SP<sub>L</sub>

(registru IFF)

Fig. 9.2 Utilizarea memoriei RAM pentru depozitarea temporară a unor informații.

Fig. 9.3 Imaginea registrelor utilizatorului.

0000—BFFF<sub>H</sub> — la dispoziția utilizatorului

C000—CFFF<sub>H</sub> — editor de texte

D000—DFFF<sub>H</sub> — asamblor

E000—E7FF<sub>H</sub> — sistem de operare (monitor) — prezentat în continuare

E800—FEFF<sub>H</sub> — la dispoziția utilizatorului

FF00—FFFF<sub>H</sub> — RAM, utilizat de monitor

Utilizarea memoriei în zona FF00—FFFF<sub>H</sub> este ilustrată în figura 9.2.

Zona FFE6—FFFF<sub>H</sub> conține imaginea registrelor utilizatorului. Registrele microprocesorului sînt încărcate cu informațiile din această zonă la lansarea în execuție a unui program al utilizatorului, prin comanda „I”. Reciproc, conținutul registrelor microprocesorului este salvat în această zonă la execuția unei comenzi de întrerupere, „B” (break-point command). Conținutul zonei poate fi modificat cu ajutorul comenzii „M”.

Figura 9.3 ilustrează utilizarea acestei zone de memorie.

O noțiune importantă utilizată este cea de canal. Un canal reprezintă o locație fixă de memorie la care este depozitată adresa unui program (driver) de lucru cu un anumit dispozitiv periferic. Cînd se execută operații de I/E, printr-un canal, monitorul utilizează adresa driver-ului depozitată în locația de memorie corespunzătoare. Pentru orice canal se poate selecta un anumit driver care corespunde unui anumit periferic, prin depunere la adresa canalului a adresei driver-ului. În plus, utilizatorul poate să-și scrie un driver pentru un anumit periferic și să-l introducă adresa la un anumit canal.



FF32	DRIVER	MSB
FF31	IESIRE SURSA	LSB
FF30	DRIVER	MSB
FF2F	INTRARE SURSA	LSB
FF2E	DRIVER	MSB
FF2D	IESIRE OBIECT	LSB
FF2C	DRIVER	MSB
FF2B	INTRARE OBIECT	LSB
FF2A	DRIVER	MSB
FF29	IESIRE CONSOLA	LSB
FF28	DRIVER	MSB
FF27	INTRARE CONSOLA	LSB

Fig. 9.4 Adresele de driver pentru canalele sursă, obiect și consolă.

efectiv utilizate doar adresele de port D0—D3<sub>H</sub>, D8—DB<sub>H</sub>, DC—DF<sub>H</sub>, F4<sub>H</sub> și F5<sub>H</sub>.

### Comenzile acceptate de monitor

Comenzile constau din identificator (o literă), operand (sau operanzi, separați de virgule sau spații) și un terminator. Inițializarea cu ajutorul butonului RESET trebuie urmată de tastarea caracterului „S” sau „↵” (retur de car), ceea ce permite sistemului măsurarea vitezei de transmisie a consolei (110, 300, 600, 1200, 2400, 4800 sau 9600 Bd). Sistemul trimite un retur de car, „↵”, avansează cu o linie, emițind „LF”, și tipărește un punct, „.”, fiind gata pentru primirea comenzilor.

Identificatorul comenzii poate fi M, P, D, L, E, H, C, B, sau R (discutate în continuare).

Operanzii sînt formați din 4 cifre hexazecimale, în cazul tastării mai multor cifre fiind considerate doar ultimele 4. Sînt permise operații aritmetice în hexazecimal (adunări și scăderi) și utilizarea de mnemonice (:MN), echivalente cu 4 cifre hexazecimale. Semnul „\$” reprezintă adresa curentă plus o unitate (se utilizează pentru salturi relative). Semnul „=” poate fi utilizat pentru afișarea rezultatului hexazecimal al unei operații. În exemplele de mai jos, caracterele subliniate sînt tastate de utilizator :

4F2A — operand 4F2A<sub>H</sub> (caracterele subliniate se tastează de către utilizator).

:PC — mnemonica PC, echivalentă cu adresa FFFE<sub>H</sub>, deci operandul FFFE<sub>H</sub>

5038—5000 — operand 0038<sub>H</sub>

5038—5000=0038 — operand 0038<sub>H</sub>

5038 — \$ — dacă adresa curentă este 5000<sub>H</sub>, atunci \$=5001<sub>H</sub>, iar operandul este 0037<sub>H</sub>

5038—\$=0037 — operand 0037<sub>H</sub>

305038 — operand 5038<sub>H</sub>, format din ultimele 4 cifre

305038=5038 — operand 5038<sub>H</sub>

Mnemonicele sînt echivalente cu adrese de 4 cifre hexazecimale, la care pot exista date de 1 octet sau de 2 octeți (marcate cu „\*”), rezultînd afișarea a 2 sau 4 cifre hexazecimale. Mnemonicele recunoscute și adresele pe care le reprezintă sînt următoarele :

:PC\*, :A, :F, :I, :IF, :B, :C, :D, :E, :H, :L, :A', :F', :B', :C', :D', :E', :H', :L', :IX\*, :IY\*, :SP\*, cu adresele date în figura 9.3 (pentru PC, IX, IY, și SP este adresa octetului inferior);



:CI\*, :CO\*, :OI\*, :OO\*, :SI\*, :SO\* (adrese de driver, date în figura 9.4, de unde se ia adresa octetului inferior)

:TK E6B3<sub>H</sub> — adresă driver pentru claviatură terminal — utilizează circuitul SIO

:TT E680<sub>H</sub> — adresă driver pentru cap imprimare terminal — utilizează circuitul SIO

:ST E67E<sub>H</sub> — adresă driver imprimantă tip Silent 700 Printer

:TR E6A5<sub>H</sub> — adresă driver cititor teletype (utilizează SIO)

:PR E6C6<sub>H</sub> — adresă driver cititor rapid de bandă — utilizează PIO

:PP E6FA<sub>H</sub> — adresă driver perforator rapid de bandă — utilizează PIO

:AS C000<sub>H</sub> — adresă apel asamblor

:ED D48B<sub>H</sub> — adresă apel editor

:LP E6F0<sub>H</sub> — adresă driver pentru imprimantă de linie — utilizează circuitul PIO

:ER D4D9<sub>H</sub> — adresă de reintrare în editor

:TI DF9B<sub>H</sub> — adresă driver intrare de bandă

:TO DF2F<sub>H</sub> — adresă driver ieșire de bandă

Terminatorul unei comenzi, notat uneori cu „t”, poate fi „↵” (determină preluarea comenzii), „^” (carat sau săgeată în sus; pentru comenzile M sau P determină afișarea locației sau portului care a fost reactualizat) și „.” (determină omiterea comenzii și așteptarea altei comenzi).

1. **Comanda M — afișarea și reactualizarea memoriei.** Permite afișarea și/sau modificarea conținutului unor locații de memorie și a registrelor microprocesorului. Formatul ei este `·Maaaat`. Monitorul afișează conținutul locației de memorie specificată. Dacă trebuie modificat, se tastează noua valoare, iar în caz contrar se încheie cu „.” sau se afișează conținutul următoarei locații. Exemple:

a. `·M 5001↵`

`5001 A3 F3FF↵` s-a modificat conținutul locației 5001<sub>H</sub> din A3<sub>H</sub>

`5002 A4. în FFH` (se consideră ultimele două cifre)

b. `·M:PC↵`

`:PC 433F7F50↵` s-a modificat conținutul registrului PC din 433F în

`0000 20. 7F50`

În funcție de caracterele introduse după afișarea conținutului unei locații de memorie, pot rezulta diferite acțiuni, după cum urmează:

1. ↵ nici un operand introdus, se afișează următoarea adresă

2. ^ nici un operand introdus, se afișează conținutul locației anterioare

3. aa. operandul aa este omis, fără modificarea conținutului locației; se iese din comanda M

4. aa. ↵ operandul aa înlocuiește conținutul anterior al locației de memorie și se afișează conținutul locației următoare

5. aa ^ operandul înlocuiește conținutul anterior al locației de memorie și afișează aceeași locație

Registrele microprocesorului (PC, A, ..., SP—fig. 9.3) pot fi inițializate înainte de execuția unui program sau examinate, de exemplu după oprirea unui program la un punct de întrerupere (breakpoint). Când se revine la execuția programului utilizatorului, noile valori sînt introduse în registrele microprocesorului.

Un exemplu de calcul al salturilor relative este prezentat în continuare:

`·M 4000+1000↵`

`5000 20 18↵`

examinează locația 5000<sub>H</sub> (calculată ca o sumă) înscrează primul octet al instrucțiunii JR calculează



5001 FS 5038--\$=0036  $\wedge$  și afișează lungimea saltului relativ de la 5001<sub>H</sub> la 5038<sub>H</sub>.  
5001 36. afișează lungimea saltului relativ

Salturile înainte trebuie să aibă o lungime maximă de 7F<sub>H</sub>, iar cele înapoi, de cel mult 80<sub>H</sub>. Valorile calculate vor fi deci în domeniile 0000—007F<sub>H</sub>, respectiv FF80—FFFE<sub>H</sub>.

Comanda „M” permite verificarea și/sau modificarea asiguărilor pentru canalele sistemului. De exemplu:

·M :CI $\curvearrowright$	intrare consolă=intrare terminal
:CI :TK $\curvearrowright$	ieșire consolă=ieșire terminal(imprimantă)
:CO :TT $\curvearrowright$	ieșire obiect=cititor de bandă perforată (după modificare)
:OI 6363 :PR $\curvearrowright$	nu se fixează un anumit driver de dispozitiv
:OO 0063 $\curvearrowright$	
:SI 6363 $\curvearrowright$	
:SO 6300 $\curvearrowright$	
FF33 80	codul 80 <sub>H</sub> indică sfârșitul tabeli de mnemonice.

Comanda M mai permite și afișarea unui bloc de memorie (cîte 16 locații pe linie). De exemplu:

·M 4100,4127  $\curvearrowright$  sau  
·M 4100 4127  $\curvearrowright$   
afișează conținutul locațiilor de memorie de la 4100, la 4127, inclusiv.

**2. Comanda P — afișarea și/sau modificarea porturilor.** Comanda permite numai modificarea porturilor care nu sînt exclusiv intrări. De asemenea, nu se pot citi porturile care sînt exclusiv ieșiri. Formatul comenzii este .P aa (aa este adresa portului). Monitorul afișează conținutul acestui port, care poate fi lăsat neschimbat sau poate fi modificat, similar cu modul utilizat pentru comanda M. Terminatorii  $\curvearrowright$ ,  $\wedge$ , aa., aa  $\curvearrowright$ , aa  $\wedge$  au aceeași semnificație ca și în cazul comenzii M, dar se referă la porturi.

Un exemplu de programare a unui port PIO cu ajutorul comenzii P este următorul:

.P D1 $\curvearrowright$	programează portul PIO1A (D0 <sub>H</sub> —date, D1 <sub>H</sub> —control) în modul I/E de bit
D1 FF CF $\wedge$	codul CF <sub>H</sub> este cuvîntul de comandă pentru modul de I/E de bit; portul D1 <sub>H</sub> este doar ieșire
D1 FF 00 $\wedge$	programează toți biții portului D0 <sub>H</sub> ca ieșiri
D1 FF $\wedge$	va afișa portul anterior
D0 D0 AA $\wedge$	înscrie valoarea AA <sub>H</sub> în portul D0 <sub>H</sub>
D0 AA	

**3. Comanda D — descarcă memoria (dump memory).** Această comandă descarcă un bloc de memorie la canalul de ieșire obiect, într-un format absolut, compatibil cu ieșirea obiect produsă de asamblor. Dacă se descarcă memoria pe bandă de hîrtie, ea poate fi citită ulterior cu ajutorul comenzii „L”. Formatul comenzii este Daaaa,bbbbt. De exemplu, D 200,220  $\curvearrowright$  descarcă memoria între locațiile 0200<sub>H</sub> și 0220<sub>H</sub>, inclusiv. Evitarea descărcării memoriei se face tastînd un punct („.”). Oprirea descărcării în timpul efectuării acesteia se face doar prin apăsarea butonului RESET. Banda perforată are 20 cm de început de bandă neperforată (8 inci),



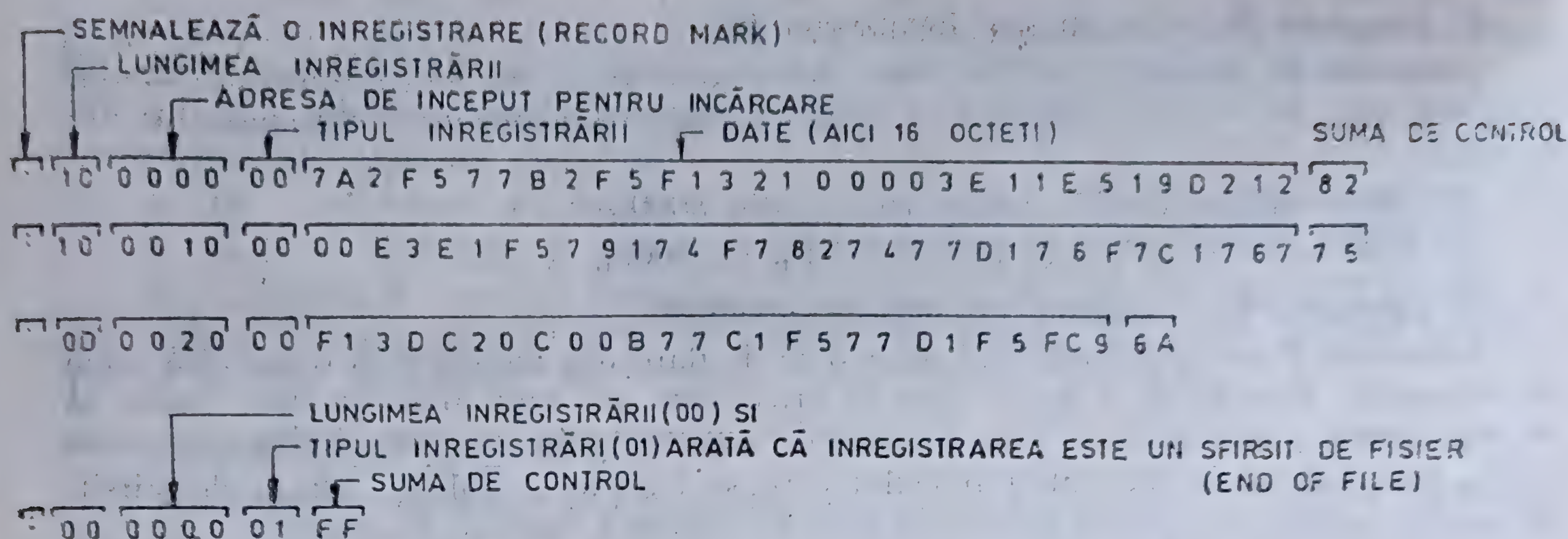


Fig. 9.5 Formatul datelor pe bandă de hîrtie perforată.

urmați de blocul de memorie și de 20 de cm de sfîrșit de bandă neperforată (zona neperforată de la început se numește leader iar cea de la sfîrșit, trailer).

Operația de „descărcare” a memoriei prin comanda D nu afectează conținutul memoriei, deci acesta rămîne neschimbat după executarea comenzii.

**4. Comanda L — încarcă memoria (load memory).** Comanda L permite încărcarea unui program absolut și/sau a unor date în memorie, prin canalul de intrare obiect. Formatul datelor trebuie să fie compatibil cu cel produs ca ieșire obiect de către asamblor.

Formatul comenzii este .L   . În cazul citirii de bandă perforată, aceasta trebuie să fie poziționată cu leader-ul pe mecanismul de citire cînd se dă comanda de încărcare, după care se activează cititorul. Citirea benzii se face la adresa indicată în cadrul înregistrării.

Dacă se găsește o eroare de paritate la încărcare, adresa de start a liniei care urmează după linia suspectă este tipărită la canalul de ieșire consolă.

Formatul utilizat pentru încărcarea benzilor în format hexazecimal absolut este cel din figura 9.5.

Zona sumei de control reprezintă complementul față de 2 al sumei tuturor octeților (în binar), înregistrați pe bandă, de la lungimea înregistrării, pînă la ultimul octet de date, inclusiv. Ca urmare, suma tuturor octeților, inclusiv suma de control, va fi 0000<sub>H</sub>.

#### 5. Comanda E — execută un program al utilizatorului.

Formatul comenzii este .E aaaa, unde aaaa este adresa programului, sau .Et, pentru transferul controlului la adresa specificată de registrul PC din zona de memorie care conține registrele utilizatorului. Înaintea executării programului, monitorul încarcă registrele microprocesorului cu informațiile conținute în această zonă.

Exemple :

a. .E 1200   

Execută programul de la adresa 1200<sub>H</sub>. Pentru revenire în monitor, trebuie inclus fie un punct de întrerupere în program, fie un salt în monitor, fie se acționează butonul de RESET.

b. .M :PC   

examinează registrul PC al utilizatorului (din memoria RAM)

:PC 62FF 1220   

înscrie 1220<sub>H</sub> în PC (din memoria RAM)

0000 20   

:E   

execută programul de la adresa 1220

Comanda E poate fi utilizată, împreună cu comanda B, pentru executarea unor porțiuni dintr-un program, în scopul depanării acestuia.



## 6. Comanda H — aritmetică hexazecimală

Comanda H permite calculul unor sume (algebrice) în hexazecimal. Formatul comenzii este  $\underline{H} + \underline{aaaa} + \underline{bbbb} + \dots + \underline{yyyy} = \underline{zzzt}$ . Două exemple sînt date în continuare:

$\underline{H} \ 5000 - 4FFF = 0001$  ↯ scade  $4FFF_H$  din  $5000_H$

$\underline{H} \ 5000 + 4FFF = 9FFF$  ↯ adună  $4FFF_H$  cu  $5000_H$

## 7. Comanda C — copiază un bloc de memorie

Comanda C permite transferul unui bloc de date din memorie în orice altă zonă de memorie. Transferul se poate face în orice sens, iar noul bloc de date poate să se suprapună peste vechiul bloc. Formatul comenzii este  $\underline{C} \ \underline{aaaa}, \underline{bbbb}, \underline{cccc}$ , unde  $\underline{aaaa}_H$  este adresa de început a blocului care se transferă,  $\underline{bbbb}_H$ , adresa de sfîrșit, iar  $\underline{cccc}_H$ , adresa de început a zonei în care se transferă datele: Exemple:

a.  $\underline{C} \ 100, 200, 1200$  ↯ blocurile nu se suprapun

b.  $\underline{C} \ 100, 200, 150$  ↯ blocurile de date se suprapun

## 8. Comanda B — punct de întrerupere (breakpoint)

Comanda B fixează o „capcană” în programul utilizatorului. La întîlnirea acesteia, controlul revine monitorului, cu posibilitatea de a verifica registrele, porturile de I/E și conținutul memoriei. Programul poate fi întrerupt numai dacă este înscris în RAM. Formatul comenzii poate fi:

a.  $\underline{B} \ \underline{aaaa}, \underline{bt}$  fixează un punct de întrerupere la adresa  $\underline{aaaa}_H$ ;  $b=0$  indică format scurt, iar  $b=1$  indică format lung pentru tipărirea registrelor (PC, A, F și respectiv toate registrele interne)

b.  $\underline{B} \ \underline{aaaat}$  fixează un punct de întrerupere la adresa  $\underline{aaaa}_H$ , cu format scurt pentru tipărirea registrelor

c.  $\underline{Bt}$  șterge orice punct de întrerupere anterior

La executarea unei comenzi B, se anulează orice comandă B, anterioară. Monitorul extrage și salvează 3 octeți din programul utilizatorului, începînd de la adresa de întrerupere ( $\underline{aaaa}_H$ ), introducînd în locul lor o instrucțiune de salt în monitor, pentru tratarea întreruperii.

După executarea comenzii B, în formatul a sau b de mai sus, utilizatorul poate executa programul înscris în memoria RAM. La întîlnirea adresei  $\underline{aaaa}_H$ , controlul revine monitorului, care reface cei 3 octeți înlocuiți anterior cu instrucțiunea de salt, salvează conținutul tuturor registrelor în zona  $FFE6-FFFF_H$ , tipărește la consolă adresa de întrerupere (PC) și valoarea registrelor A și F (format scurt) sau toate registrele interne (format lung), după care revine la modul de acceptare a unor noi comenzi.

După aplicarea unui semnal RESET, cei 3 octeți de salt introduși în programul utilizatorului pot fi înlocuiți și manual, de la consolă.

Adresa de întrerupere poate fi fixată numai la începutul unora dintre instrucțiunile din programul utilizatorului. Și în acest caz trebuie analizate acțiunile care pot să apară. De exemplu, pentru secvența de instrucțiuni de mai jos, punctul de întrerupere nu se poate plasa de la adresa I.2, pentru că în cazul executării unui salt la adresa I.3 (instrucțiunea de la adresa I.1), controlul programului ar fi transferat la al treilea octet al instrucțiunii de salt introduse în program prin comandă B:

I.1: JR NZ, I.3--\$

I.2: LD A, 00H ; INSTRUCTIUNE DE 2 OCTETI

I.3: LD B, 0FH ; INSTRUCTIUNE DE 2 OCTETI

Dacă utilizatorul încearcă să fixeze în ROM un punct de întrerupere, eroarea nu este semnalată în nici un mod.



La întâlnirea unui punct de întrerupere, monitorul salvează starea întreruperilor (prin IFF), în registrul :IF din RAM, starea întreruperilor fiind refăcută, conform cu conținutul lui :IF, când controlul se transferă la programul utilizatorului.

### 9. Comanda R — afișează registrele CPU la consolă

Formatul comenzii poate fi:

a. :Rt — tipărește conținutul tuturor registrelor microprocesorului.

b. :R lt — tipărește și numele tuturor registrelor.

Formatul lui F este cel cunoscut: registrul IF conține în bitul 3 valoarea lui IFF (deci IF=00<sub>H</sub> pentru IFF=0 și IF=04<sub>H</sub> pentru IFF=1); IFF=0 arată că nu se acceptă întreruperi (de exemplu după RESET, IFF=0).

### Posibilități de intrare/ieșire

Monitorul are 3 canale de I/E la care pot fi asignate diferite dispozitive periferice. Canalul pentru consolă este utilizat pentru a primi comenzi și pentru a răspunde la ele, ca și pentru manipularea informațiilor de editare și control. Canalul obiect este utilizat de comenzile L și D pentru citirea sau scrierea de date în cod obiect. Canalul sursă este utilizat numai de asamblor și de editorul de texte pentru a citi sau scrie date în limbaj de asamblare (programe sursă). Asignările pentru aceste canale sînt depozitate în memoria RAM și pot fi reactualizate cu comanda M. Driver-urile disponibile sînt :TK, :TR, :TT, :PR, :PP, :LP. La alimentarea sistemului, sau după acționarea butonului de RESET, asignările făcute de monitor sînt :TK pentru CI și :TT pentru CO. Asignările pentru canalele obiect și sursă trebuie efectuate de către utilizator dacă acestea sînt necesare. Configurația lor nu este modificată la RESET.

Este posibilă adăugarea de noi drivere de I/E și definirea de mnemonice pentru ele, avînd în vedere următoarele elemente:

- registrul E este registrul de control iar registrul D este registrul de date
- în cadrul registrului E, bitul E<sub>7</sub>=1 semnifică modul de I/E cu „revenire imediată”: dacă datele sau dispozitivul sînt gata, efectuează operația de I/E și șterge în continuare bitul E<sub>7</sub>; în caz contrar, bitul E<sub>7</sub>=1 în continuare; bitul E<sub>7</sub>=0 semnifică așteptarea pînă la efectuarea operației de I/E, înainte de revenirea din driver-ul de I/E; bitul E<sub>3</sub>=1 indică inițializarea dispozitivului (se face de obicei la început și doar o singură dată); bitul E<sub>3</sub> este șters la ieșirea din drivere.
- datele de ieșire sînt conținute în registrul D, iar datele de intrare în D și A; la efectuarea operației de ieșire, conținutul lui A se pierde
- toate driver-urile disponibile: TK, TT, ST, TR, PR, PP, LP asigură revenirea imediată dacă E<sub>7</sub>=1.

### Subrutine ale monitorului, care pot fi apelate de utilizator

Condițiile de intrare și de ieșire pentru cîteva subrutine prezentate în continuare nu reflectă complet modul în care acestea pot afecta programul utilizatorului. Un examen amănunțit al codului obiect al acestora poate fi util pentru înțelegerea subrutinelor.

1. RDCHR — citește un caracter ASCII; are adresa E522<sub>H</sub>; la intrare, registrul E desemnează canalul de I/E: E=0, 1 — canalul consolă; E=2, 3 — canalul obiect; E=4, 5 — canalul sursă; dacă E<sub>7</sub>=1, se revine imediat dacă data nu este gata, în cazul driver-elor care acceptă revenirea imediată; la ieșire, E este neschimbat, cu excepția lui E<sub>7</sub>, care va fi 0 dacă s-a citit data; registrele D și A conțin caracterul ASCII; bitul 7 al datei (bitul de paritate) este șters; subrutina apelează driver-ul de I/E specificat în registrul IF.

2. WRCHR — scrie un caracter ASCII; are adresa E527<sub>H</sub>; la intrare, registrul E desemnează canalul de I/E: E=0, 1 — canalul consolă; E=2, 3 — canalul obiect; E=4, 5 — canalul sursă; dacă E<sub>7</sub>=1, se revine imediat dacă dispozitivul de I/E nu este gata pentru date, în cazul driver-elor care acceptă revenirea imediată; D conține data care trebuie scrisă; la ieșire, E este neschimbat, doar E<sub>7</sub>=0 dacă data a



fost scrisă; D conține data de ieșire; A este distrus; apelează driver-ul de I/E conectat la canalul de I/E specificat în registrul E.

3. PACC — tipărește conținutul registrului A; are adresa E58B<sub>H</sub>; la intrare, E desemnează canalul de I/E, ca mai sus; nu este valabilă facilitatea de revenire imediată; A conține echivalentul binar al celor 2 cifre hexazecimale care vor fi tipărite în ASCII; la ieșire, conținutul lui A este distrus; rutinele apelate sînt PRVAL și WRCHR.

4. PRVAL — convertește cei 4 biți mai puțin semnificativi (o cifră hexazecimală) ai registrului A în ASCII; are adresa E5AF<sub>H</sub>; la intrare, A conține în jumătatea inferioară cifra hexazecimală care va fi convertită în ASCII; la ieșire, D și A conțin codul ASCII obținut.

5. ECHO — citește și scrie un caracter prin același canal de I/E; are adresa E597<sub>H</sub>; la intrare, E desemnează canalul de intrare/ieșire, ca mai sus, fără posibilitatea de revenire imediată; la ieșire, D conține caracterul citit și imprimat; conținutul lui A este distrus; apelează RDCHR și WRCHR.

6. CRLF — trimite la dispozitivul de I/E un retur de car (CR=↵) și un avans de linie (LF); are adresa E59C<sub>H</sub>; la intrare, E desemnează canalul de I/E, ca mai sus, fără posibilitatea de revenire imediată; la ieșire, D conține reprezentarea ASCII pentru LF (0A<sub>H</sub>), iar conținutul lui A este distrus; apelează WRCHR.

7. SPACE — trimite la canalul de I/E un spațiu (blank); are adresa E5A5<sub>H</sub>; la intrare, E desemnează canalul de I/E, ca mai sus; la ieșire, D conține codul ASCII pentru spațiu (20<sub>H</sub>), iar conținutul lui A este distrus; apelează WRCHR.

8. PTXT — imprimă un șir dintr-un text; are adresa E3C7<sub>H</sub>; la intrare, E desemnează canalul de I/E, ca mai sus, fără posibilitatea de revenire imediată; HL conține adresa de început a textului ASCII; ultimul caracter al textului trebuie să fie ETX (codul ASCII 03<sub>H</sub>), care nu este imprimat; la ieșire, D conține codul 03<sub>H</sub>, HL conține adresa de memorie la care se află caracterul ETX, iar conținutul lui A este distrus; apelează WRCHR.

9. ASBIN — convertește reprezentarea ASCII a unei cifre hexazecimale, în binar; are adresa E583<sub>H</sub>; la intrare, A conține caracterul ASCII care se va converti, iar la ieșire, A conține rezultatul conversiei; nu se face nici o verificare la erori.

10. RENTRY — adresa de reintrare în monitor, E11D<sub>H</sub>; se efectuează salt la această adresă (nu se apelează ca subrutină); monitorul va trimite un retur de car, un avans de linie și va tipări un punct (.,.); registrele utilizatorului nu sînt salvate; monitorul este în continuare gata să accepte altă comandă.

### Driveri care pot fi apelate de utilizator

1. TK — citește un caracter de la claviatura terminalului; utilizează SIO; are adresa E6B3<sub>H</sub>; la intrare, E<sub>7</sub>=1 semnifică revenire imediată dacă nu este gata caracterul; la ieșire, E<sub>7</sub>=0 dacă a fost citit caracterul; D și A conțin caracterul ASCII, cu bitul de paritate șters.

2. TT — imprimă un caracter la terminal sau la imprimantă; utilizează SIO; are adresa E680<sub>H</sub>; la intrare, E<sub>7</sub>=1 semnifică revenire imediată dacă dispozitivul nu este gata pentru a primi caracterul; registrul D conține caracterul de imprimat; la ieșire, E<sub>7</sub>=0, dacă data a fost imprimată; bitul E<sub>4</sub> este utilizat intern și este întotdeauna șters la ieșire; registrul D conține caracterul imprimat, iar conținutul registrului A este distrus.

3. ST — la fel cu TT, cu excepția inserării unei întârzieri cînd se trimite la ieșire un retur de car și un avans de linie (CRLF); utilizează SIO; are adresa E67E<sub>H</sub>.

4. TR — la fel cu TK, cu excepția faptului că se trimite la ieșire un semnal de avans al cititorului, pentru a aduce următorul caracter de pe bandă; utilizează SIO; are adresa E6A5<sub>H</sub>.

5. PR — citește un caracter de la un cititor rapid de bandă; utilizează PIO; portul A date (D0<sub>H</sub>) și portul A control (D1<sub>H</sub>); întreruperile trebuie să fie progra-



mate în modul 2 (IM2); are adresa E6C6<sub>H</sub>; la intrare, bitul E<sub>3</sub>=1 arată că dispozitivul trebuie inițializat; bitul E<sub>7</sub>=1 semnifică revenire imediată; la ieșire, E<sub>3</sub>=0; registrele D și A conțin caracterul ASCII citit, cu bitul de paritate șters; întreruperile au fost folosite și sînt validate la ieșire.

6. PP — trimite un caracter la un perforator rapid de bandă; utilizează PIO: portul B date (D2<sub>H</sub>) și portul B control (D3<sub>H</sub>); întreruperile trebuie să fie programate în modul 2 (IM2); are adresa E6FA<sub>H</sub>; la intrare, E<sub>3</sub>=1 arată că dispozitivul trebuie inițializat; registrul D conține caracterul ASCII care trebuie trimis la ieșire; E<sub>7</sub>=1 semnifică revenirea imediată; la ieșire, E<sub>3</sub>=0; registrul D conține data care a fost trimisă la ieșire; întreruperile au fost utilizate și sînt validate la ieșire.

7. LP — trimite un caracter la o imprimantă de linie; utilizează PIO: portul B date (D6<sub>H</sub>) și portul B control (D7<sub>H</sub>); întreruperile trebuie să fie în modul 2 (IM2); are adresa E6F0<sub>H</sub>; la intrare, E<sub>3</sub>=1 arată că dispozitivul trebuie inițializat; E<sub>7</sub>=1 semnifică revenire imediată; registrul D conține caracterul ASCII care trebuie trimis la ieșire; la ieșire, E<sub>3</sub>=0; registrul D conține data de ieșire; întreruperile au fost utilizate și sînt validate la ieșire.

Programul monitor, care ocupă zona de memorie E000<sub>H</sub>—E7FF<sub>H</sub> (2 kiloocteți) și care este în mod obișnuit înscris în memorie EPROM, este listat în continuare.

E000	C3	03	E0	DB	F4	21	90	FF	22	E6	FF	AF	32	0C	FF	3E
E010	05	D3	D8	3E	03	D3	DE	0E	F5	11	01	00	ED	78	F2	1C
E020	E0	13	ED	78	FA	21	80	E6	FE	20	20	03	21	7E	E6	22
E030	F9	3B	F1	D3	D8	21	80	E6	FE	20	20	03	21	7E	E6	22
E040	29	FF	FE	57	3E	7C	28	02	3E	74	67	00	C3	39	E2	0D
E050	00	01	1A	00	02	34	00	04	68	00	08	D0	00	10	A0	01
E060	20	FF	7F	57	FF	13	C3	69	E0	ED	73	E6	FF	31	00	00
E070	F5	F5	B7	DB	DF	ED	57	F5	F3	C5	D5	E5	D9	08	F5	C5
E080	D5	E5	3A	FA	FF	E6	04	32	FA	FF	D9	08	DD	E5	FD	E5
E090	30	16	21	0F	FF	ED	5B	0D	FF	01	03	00	ED	D0	2A	0D
E0A0	FF	3E	81	32	0C	FF	18	09	ED	7B	E6	FF	E1	ED	73	E6
E0B0	FF	22	FE	FF	31	E6	FF	C3	21	E6	ED	73	E6	FF	31	00
E0C0	00	F5	F5	37	18	AD	3E	00	18	02	3E	01	32	12	FF	ED
E0D0	7B	E6	FF	2A	FE	FF	E5	2A	FC	FF	E5	ED	73	B4	FF	31
E0E0	E8	FF	FD	E1	DD	E1	D9	08	E1	D1	C1	F1	D9	08	E1	D1
E0F0	C1	F1	ED	47	ED	7B	B4	FF	3A	12	FF	EA	03	E1	D3	DF
E100	F1	F3	C9	D3	DF	F1	FB	C9	21	B3	E6	22	27	FF	21	47
E110	E1	22	1F	FF	AF	32	FA	FF	3E	80	32	33	FF	31	E6	FF
E120	CD	96	E6	DB	DC	ED	5E	1E	00	CD	9C	E5	11	00	2E	CD
E130	27	E5	CD	97	E5	FE	2E	28	EE	21	1C	FF	72	CD	A5	E5
E140	CD	14	E4	2A	1F	FF	E9	00	00	00	00	00	00	00	00	00
E150	3A	1C	FF	2A	16	FF	ED	5B	14	FF	FE	4D	C2	FA	E1	EB
E160	1E	01	3A	1A	FF	FE	01	28	2C	E5	C1	E5	2A	16	FF	B7
E170	ED	42	01	0F	00	B7	ED	42	06	10	28	06	D2	84	E1	7D
E180	80	47	CB	EB	E1	CD	04	E6	CD	6E	E6	CD	9C	E5	CB	6B
E190	C2	2C	E1	18	D4	CD	F7	E5	CB	73	28	12	E5	D9	DD	E1
E1A0	DD	6E	00	DD	66	01	1E	00	CD	F7	E5	D9	18	04	7E	CD
E1B0	AA	E5	D5	22	1D	FF	CD	14	E4	2A	1D	FF	D1	3A	1B	FF
E1C0	FE	2E	CA	2C	E1	3A	1A	FF	E6	03	28	20	3A	14	FF	77
E1D0	23	CB	73	28	05	3A	15	FF	77	23	3A	1B	FF	FE	0D	28
E1E0	B4	2B	2B	CD	BA	E5	FA	EA	E1	23	18	A9	3A	1B	FF	FE
E1F0	5E	28	EE	23	CB	73	20	F1	18	9B	FE	50	20	38	3A	14
E200	FF	4F	1E	01	79	CD	AA	E5	ED	78	CD	AA	E5	C5	CD	14
E210	E4	C1	3A	1B	FF	FE	2E	CA	2C	E1	67	3A	1A	FF	A7	28
E220	0D	3A	14	FF	ED	79	3E	5E	BC	28	D7	0C	18	D4	3E	5E



mate în modul 2 (IM2); are adresa E6C6<sub>H</sub>; la intrare, bitul E<sub>3</sub>=1 arată că dispozitivul trebuie inițializat; bitul E<sub>7</sub>=1 semnifică revenire imediată; la ieșire, E<sub>3</sub>=0; registrele D și A conțin caracterul ASCII citit, cu bitul de paritate șters; întreruperile au fost folosite și sînt validate la ieșire.

6. PP — trimite un caracter la un perforator rapid de bandă; utilizează PIO: portul B date (D2<sub>H</sub>) și portul B control (D3<sub>H</sub>); întreruperile trebuie să fie programate în modul 2 (IM2); are adresa E6FA<sub>H</sub>; la intrare, E<sub>3</sub>=1 arată că dispozitivul trebuie inițializat; registrul D conține caracterul ASCII care trebuie trimis la ieșire; E<sub>7</sub>=1 semnifică revenirea imediată; la ieșire, E<sub>3</sub>=0; registrul D conține data care a fost trimisă la ieșire; întreruperile au fost utilizate și sînt validate la ieșire.

7. LP — trimite un caracter la o imprimantă de linie; utilizează PIO: portul B date (D6<sub>H</sub>) și portul B control (D7<sub>H</sub>); întreruperile trebuie să fie în modul 2 (IM2); are adresa E6F0<sub>H</sub>; la intrare, E<sub>3</sub>=1 arată că dispozitivul trebuie inițializat; E<sub>7</sub>=1 semnifică revenire imediată; registrul D conține caracterul ASCII care trebuie trimis la ieșire; la ieșire, E<sub>3</sub>=0; registrul D conține data de ieșire; întreruperile au fost utilizate și sînt validate la ieșire.

Programul monitor, care ocupă zona de memorie E000<sub>H</sub>—E7FF<sub>H</sub> (2 kiloocteți) și care este în mod obișnuit înscris în memorie EPROM, este listat în continuare.

E000	C3	03	E0	DB	F4	21	90	FF	22	E6	FF	AF	32	0C	FF	3E
E010	05	D3	D8	3E	03	D3	DE	0E	F5	11	01	00	ED	78	F2	1C
E020	E0	13	ED	78	FA	21	E0	31	4E	E0	33	E1	37	ED	52	38
E030	F9	3B	F1	D3	D8	21	80	E6	FE	20	20	03	21	7E	E6	22
E040	29	FF	FE	57	3E	7C	28	02	3E	74	67	00	C3	39	E2	0D
E050	00	01	1A	00	02	34	00	04	68	00	08	D0	00	10	A0	01
E060	20	FF	7F	57	FF	13	C3	69	E0	ED	73	E6	FF	31	00	00
E070	F5	F5	B7	DB	DF	ED	57	F5	F3	C5	D5	E5	D9	08	F5	C5
E080	D5	E5	3A	FA	FF	E6	04	32	FA	FF	D9	08	DD	E5	FD	E5
E090	30	16	21	0F	FF	ED	5B	0D	FF	01	03	00	ED	D0	2A	0D
E0A0	FF	3E	81	32	0C	FF	18	09	ED	7B	E6	FF	E1	ED	73	E6
E0B0	FF	22	FE	FF	31	E6	FF	C3	21	E6	ED	73	E6	FF	31	00
E0C0	00	F5	F5	37	18	AD	3E	00	18	02	3E	01	32	12	FF	ED
E0D0	7B	E6	FF	2A	FE	FF	E5	2A	FC	FF	E5	ED	73	B4	FF	31
E0E0	E8	FF	FD	E1	DD	E1	D9	08	E1	D1	C1	F1	D9	08	E1	D1
E0F0	C1	F1	ED	47	ED	7B	B4	FF	3A	12	FF	EA	03	E1	D3	DF
E100	F1	F3	C9	D3	DF	F1	FB	C9	21	B3	E6	22	27	FF	21	47
E110	E1	22	1F	FF	AF	32	FA	FF	3E	80	32	33	FF	31	E6	FF
E120	CD	96	E6	DB	DC	ED	5E	1E	00	CD	9C	E5	11	00	2E	CD
E130	27	E5	CD	97	E5	FE	2E	28	EE	21	1C	FF	72	CD	A5	E5
E140	CD	14	E4	2A	1F	FF	E9	00	00	00	00	00	00	00	00	00
E150	3A	1C	FF	2A	16	FF	ED	5B	14	FF	FE	4D	C2	FA	E1	EB
E160	1E	01	3A	1A	FF	FE	01	28	2C	E5	C1	E5	2A	16	FF	B7
E170	ED	42	01	0F	00	B7	ED	42	06	10	28	06	D2	84	E1	7D
E180	80	47	CB	EB	E1	CD	04	E6	CD	6E	E6	CD	9C	E5	CB	6B
E190	C2	2C	E1	18	D4	CD	F7	E5	CB	73	28	12	E5	D9	DD	E1
E1A0	DD	6E	00	DD	66	01	1E	00	CD	F7	E5	D9	18	04	7E	CD
E1B0	AA	E5	D5	22	1D	FF	CD	14	E4	2A	1D	FF	D1	3A	1B	FF
E1C0	FE	2E	CA	2C	E1	3A	1A	FF	E6	03	28	20	3A	14	FF	77
E1D0	23	CB	73	28	05	3A	15	FF	77	23	3A	1B	FF	FE	0D	28
E1E0	B4	2B	2B	CD	BA	E5	FA	EA	E1	23	18	A9	3A	1B	FF	FE
E1F0	5E	28	EE	23	CB	73	20	F1	18	9B	FE	50	20	38	3A	14
E200	FF	4F	1E	01	79	CD	AA	E5	ED	78	CD	AA	E5	C5	CD	14
E210	E4	C1	3A	1B	FF	FE	2E	CA	2C	E1	67	3A	1A	FF	A7	28
E220	0D	3A	14	FF	ED	79	3E	5E	BC	28	D7	0C	18	D4	3E	5E



E230	BC	20	F8	0D	18	CC	C3	57	E2	3E	04	D3	DD	7C	D3	DD
E240	3E	05	D3	DD	3E	6A	D3	DD	3E	03	D3	DD	3E	E1	D3	DD
E250	C3	08	E1	FF	FF	FF	FF	FE	42	20	43	D5	3A	0C	FF	A7
E260	28	10	ED	5B	0D	FF	21	0F	FF	01	03	00	ED	B0	AF	32
E270	0C	FF	3A	1A	FF	A7	E1	28	1F	22	0D	FF	E5	11	0F	FF
E280	01	03	00	C5	ED	B0	21	9B	E2	C1	D1	ED	B0	3A	16	FF
E290	32	13	FF	3E	01	32	0C	FF	C3	2C	E1	C3	BA	E0	FE	45
E2A0	20	0D	3A	1A	FF	A7	28	04	ED	53	FE	FF	C3	C6	E0	FE
E2B0	44	20	65	AF	ED	52	23	E5	1E	0B	CD	7B	E3	16	3A	CD
E2C0	27	E5	E1	01	10	00	AF	ED	42	30	09	09	85	47	2E	00
E2D0	28	28	18	01	41	E5	0E	00	78	CD	11	E3	2A	14	FF	7C
E2E0	CD	11	E3	7D	CD	11	E3	AF	CD	11	E3	7E	CD	11	E3	23
E2F0	10	F9	CD	76	E6	22	14	FF	18	C3	06	03	AF	4F	CD	11
E300	E3	10	F9	3E	01	CD	11	E3	CD	76	E6	CD	7B	E3	C3	27
E310	E1	F5	81	4F	F1	C3	8B	E5	FE	4C	20	6B	1E	0A	3A	1A
E320	FF	A7	C2	59	DA	CD	22	E5	D6	3A	20	F9	4F	CD	62	E3
E330	47	CD	62	E3	67	CD	62	E3	6F	CD	62	E3	3D	F5	28	07
E340	CD	62	E3	77	23	10	F9	CD	62	E3	AF	81	28	0C	1E	00
E350	7C	CD	8B	E5	7D	CD	8B	E5	1E	02	F1	20	C8	CD	22	E5
E360	18	AC	CD	22	E5	CD	83	E5	07	07	07	07	C5	4F	CD	22
E370	E5	CD	83	E5	B1	C1	F5	81	4F	F1	C9	C5	06	50	16	00
E380	CD	27	E5	10	FB	C1	C9	FE	43	20	26	B7	ED	52	23	44
E390	4D	EB	ED	5B	18	FF	B7	ED	52	30	0E	2A	18	FF	09	2B
E3A0	54	5D	2A	16	FF	ED	B8	18	05	2A	14	FF	ED	B0	C3	2C
E3B0	E1	FE	52	20	F9	3A	14	FF	A7	28	06	21	D2	E3	CD	C7
E3C0	E3	CD	5C	E6	C3	27	E1	56	7A	FE	03	C8	CD	27	E5	23
E2D0	18	F5	20	50	43	20	20	20	41	46	20	20	49	20	49	46
E3E0	20	20	42	43	20	20	20	44	45	20	20	20	48	4C	20	20
E3F0	41	27	46	27	20	42	27	43	27	20	44	27	45	27	20	48
E400	27	4C	27	20	20	49	58	20	20	20	49	59	20	20	20	53
E410	50	0D	0A	03	AF	21	14	FF	E5	FD	E1	77	01	05	00	11
E420	15	FF	ED	B0	5F	32	1A	FF	D9	57	06	00	AF	67	6F	5A
E430	D9	CD	97	E5	D9	57	CD	6A	E5	7A	20	0E	CD	83	E5	29
E440	29	29	29	85	6F	30	E8	24	18	E5	FE	24	20	06	2A	1D
E450	FF	23	18	DB	D5	E5	FD	E5	D1	EB	AF	80	28	09	7E	93
E460	77	23	7E	9A	77	18	07	7E	83	77	23	7E	8A	77	E1	D1
E470	7A	FE	2B	20	04	06	00	18	B3	FE	2D	20	04	06	01	18
E480	AB	FE	20	28	0C	FE	2C	28	08	FE	0D	28	04	FE	5E	20
E490	3B	3A	1A	FF	FE	03	28	0C	3C	32	1A	FF	FE	03	28	04
E4A0	FD	23	FD	23	7A	D9	FE	0D	28	0D	FE	5E	28	04	D9	C3
E4B0	2A	E4	16	0D	CD	27	E5	CD	A1	E5	D9	7B	A7	20	07	3A
E4C0	1A	FF	3D	32	1A	FF	7A	32	1B	FF	D9	C9	FE	3A	28	1F
E4D0	FE	2E	20	07	AF	32	1C	FF	D9	18	D7	FE	3D	20	F5	FD
E4E0	7E	01	D9	CD	8B	E5	FD	7E	00	CD	8B	E5	D9	18	90	D9
E4F0	CD	97	E5	D9	6F	D9	CD	97	E5	D9	57	5F	FE	2E	28	D4
E500	CD	6A	E5	28	02	16	20	62	CD	47	E5	28	0B	CD	41	E5
E510	28	06	AF	67	6F	C3	2F	E4	7B	CD	6A	E5	53	28	F6	C3
E520	54	E4	7B	E6	FE	18	03	7B	F6	01	E5	21	3F	E5	E5	E6
E530	07	CB	27	C6	27	6F	AF	CE	FF	67	7E	23	66	6F	E9	E1
E540	C9	D5	11	5C	E7	18	04	D5	11	33	FF	EB	7E	FE	80	28
E550	0F	BB	20	10	23	7E	E6	7F	BA	20	0A	23	5E	23	56	AF
E560	E1	EB	B7	C9	23	23	23	23	18	E2	FE	27	28	10	FE	30
E570	38	0E	FE	3A	38	08	FE	40	38	06	FE	5B	30	02	AF	C9
E580	AF	3C	C9	D6	30	FE	0A	F8	D6	07	C9	F5	0F	0F	0F	0F
E590	CD	AF	E5	F1	C3	AF	E5	CD	22	E5	18	07	16	0D	CD	27



E5A0	E5	16	0A	18	02	16	20	C3	27	E5	CD	8B	E5	18	16	16
E5B0	0F	C6	90	27	CE	40	27	57	18	ED	FD	21	33	1F	06	02
E5C0	18	04	FD	21	5C	E7	0E	00	FD	7E	00	FE	80	28	24	FD
E5D0	23	FD	23	FD	7E	00	BD	20	14	FD	23	FD	7E	00	8C	20
E5E0	0E	FD	7E	FD	47	FD	7E	FE	4F	E6	80	3C	C9	FD	23	FD
E5F0	23	18	D3	10	CD	AF	C9	CB	B3	CD	BA	E5	F2	01	E6	CB
E600	F3	C2	0F	E6	7C	CD	8B	E5	7D	CD	8B	E5	C3	A5	E5	16
E610	3A	CD	27	E5	50	CD	27	E5	51	CD	27	E5	CD	A5	E5	18
E620	EB	1E	01	CD	54	E6	3A	0C	FF	A7	3E	00	32	0C	FF	FA
E630	27	E1	3A	16	FF	3D	28	03	3D	18	11	1E	00	CD	22	E5
E640	3E	2E	BA	CA	27	E1	3E	0D	BA	20	F0	AF	F5	CD	9C	E5
E650	F1	C3	50	E2	3A	13	FF	1F	06	02	30	02	06	0D	21	FF
E660	FF	7E	CD	8B	E5	2B	7E	CD	AA	E5	2B	10	F4	C9	7E	CD
E670	AA	E5	23	10	F9	C9	97	91	CD	11	E3	C3	9C	E5	CB	E3
E680	18	23	CB	57	20	05	CB	7B	C0	18	F5	7A	D3	DC	CB	63
E690	28	0E	FE	0D	20	0A	C5	01	87	74	0D	20	FD	10	F8	C1
E6A0	CB	BB	CB	A3	C9	AF	D3	DD	DB	DD	18	D6	AF	D3	DD	DB
E6B0	DD	18	02	18	F7	CB	47	20	05	CB	7B	C0	18	F5	DB	DC
E6C0	CB	BB	CB	BF	57	C9	E5	C5	01	D1	00	3E	4F	21	21	FF
E6D0	CB	5B	28	05	CD	2E	E7	DB	D0	FB	CB	7E	20	06	CB	7B
E6E0	20	0C	18	F6	36	00	DB	D0	2F	CB	BB	CB	BF	57	18	32
E6F0	E5	21	23	FF	C5	01	D7	04	18	08	E5	21	22	FF	C5	01
E700	D3	02	F5	CB	5B	28	07	3E	0F	CD	2E	E7	18	0B	FB	CB
E710	7E	20	06	CB	7B	20	0A	18	F6	36	00	7A	0B	ED	79	CB
E720	BB	F1	C1	E1	CB	9B	C9	FF	42	E7	4C	E7	54	E7	F3	ED
E730	79	3E	28	80	ED	79	3E	83	ED	79	36	00	3E	E7	ED	47
E740	FB	C9	F5	3E	FF	32	21	FF	F1	FB	EC	4D	F5	3E	FF	32
E750	22	FF	18	F4	F5	3E	FF	32	23	FF	18	EC	41	20	FD	FF
E760	46	20	FC	FF	48	20	F5	FF	4C	20	F4	FF	42	20	F9	FF
E770	43	20	F8	FF	44	20	F7	FF	45	20	F6	FF	49	D8	EA	FF
E780	49	D9	E8	FF	49	20	FB	FF	49	46	FA	FF	53	D0	E6	FF
E790	50	C3	FE	FF	41	27	F3	FF	46	27	F2	FF	48	27	ED	FF
E7A0	4C	27	EC	FF	42	27	F1	FF	43	27	F0	FF	44	27	EF	FF
E7B0	45	27	EE	FF	43	C9	27	FF	43	CF	29	FF	4F	C9	2B	FF
E7C0	4F	CF	2D	FF	53	C9	2F	FF	53	CF	31	FF	54	4B	B3	E6
E7D0	54	54	80	E6	53	54	7E	E6	50	52	C6	E6	50	50	FA	E6
E7E0	54	52	A5	E6	41	53	00	C0	45	44	8B	D4	4C	50	F0	E6
E7F0	45	52	D9	D4	54	49	9B	DF	54	4F	2F	DF	80	00	00	BF

## 9.2. INTERPRETOR BASIC DE 12 Ko PENTRU MICROPROCESORUL Z80

Interpretorul Basic, cu o lungime de aproximativ 12 kiloocteti, ocupă spațiul de memorie 0300<sub>H</sub>—3151<sub>H</sub>. Spațiul 0000<sub>H</sub>—02FF<sub>H</sub> trebuie să fie constituit din memorie RAM, utilizată de interpretor.

Legăturile interpretorului cu sistemul de operare sînt asigurate de următoarele instrucțiuni, înscrise la adresele menționate:

ADRESA	CODUL	ETICHETA	COD OPERAȚIE	COMENTARIU
0300	C3C92F	BASIC:	JMP INIT ;	
0303	C37304	REST:	JMP RECOVER ;	
0306	C3440A	USR:	JMP ERROR ;	SEMNALIZEAZA ERORILE
0309	C33000	CI:	JMP CIN ;	INTRARE CONSOLA
030C	C31DE1	RI:	JMP RIV ;	INTRARE CITITOR



030F	C34000	CO: JMP CON ; IESIRE CONSOLA
0312	C31DE1	PO: JMP WRTV ; IESIRE PERFORATOR
0315	C34000	LO: JMP LISTX ; IESIRE DE LISTARE
0318	C30300	CSTS: JMP CSTSX ; STAREA CONSOLEI
031B	C31400	IOCHK: JMP IOCHK ; TESTAREA CONFIGURATIEI DE ; I/E
031E	C31500	IOSET: JMP IOSTX ; MODIFICARE DE I/E
0321	C32000	MEMSIZ: JMP MEMCK ; TESTAREA SFIRSITULUI MEMO- ; RIEI
0324	C31DE1	TRAP: JMP TRAPX ; PUNCT DE INTRERUPERE ; (BREAK-POINT)

Rutinele menționate mai sus sînt descrise sumar în continuare:

CIN — aduce în acumulator un caracter de la consolă  
RIV — aduce în acumulator un caracter de la cititor  
CON — în registrul C este pregătit un caracter pentru ieșire la consolă  
WRTV — ieșirea unui caracter, prin registrul C, la perforatorul de bandă (PUNCH)  
LISTX — ieșirea unui caracter prin registrul C, la dispozitivul de listare, de exemplu pentru ieșirea programelor Basic  
CSTXS — starea consolei; furnizează în A valoarea FF<sub>H</sub> cînd s-a recepționat un caracter și 00<sub>H</sub>, în caz contrar  
IOCHX — în registrul A se va da configurația actuală; poate fi scurtcircuitată prin XRA A,RET  
IOSTX — noul octet de I/E se va da în registrul C; poate fi de asemenea scurtcircuitată  
MEMCK — indică valoarea care este utilizată dacă la întrebarea relativă la mărimea memoriei, care apare după start, se răspunde cu CR  
TRAPX — începutul programului monitor propriu.

#### Descrierea comenzilor interpretorului Basic de 12 kiloocteți Programe generale utilitare

AUTO — prin comanda AUTO se determină generarea automată a numerelor de linie; în plus, se pot preciza o nouă linie de start și lungimea pasului, de exemplu: AUTO 100 sau AUTO 100,3

CLEAR — toate variabilele sînt șterse; dacă se dă suplimentar un număr, locul șirului se va considera de la această valoare; CLEAR 200 va șterge toate variabilele fixînd adresa comenzii pentru șiruri de la 200

CONTINUE — dacă s-a oprit rularea programului cu CTRL-C, se poate continua programul cu această comandă, cînd nu s-au adus schimbări la program

DELETE — o serie de linii pot fi șterse; DELETE 100-135 șterge toate liniile de la 100 pînă la 135 inclusiv

KILL — după această comandă se poate reutiliza spațiul de memorie neutilizat de matrice; KILL A,B elimină spațiul de memorie întrebuințat de matricele A și B.

LOAD — încarcă un program de la cititor; pentru aceasta se utilizează înainte NEW; LOAD P încarcă un program care a fost memorat cu SAVE P; pentru aceasta se admite ca nume doar o literă; LOAD P efectuează o citire de testare; dacă se încarcă data, atunci se trimite la ieșire caracterul ASCII „Bell”

LOADGO — ca și LOAD, dar cu pornirea programului; LOADGO P,100 încarcă programul P și îl începe de la 100



NEW — șterge întreaga memorie de lucru (programul Basic și toate variabilele)

PRECISION — precizia de calcul prestabilită este de 11 cifre; PRECISION 4 determină ca numerele editate să fie de 4 cifre; calculele se efectuează intern tot cu 11 cifre, dar rezultatul este rotunjit la 4 cifre.

RENUMBER — numerotare nouă; toate referințele și salturile sînt automat fixate; RENUMBER numerotează toate liniile cu pași de cîte 10, începînd de la 10; RENUMBER 110 începe de la 110; RENUMBER 120,3 numerotează începînd cu 120, cu pași egali cu 3; RENUMBER 500,5,300 începe cu rîndul 300, pînă la 500, cu pași egali cu 5 (în acest mod se pot completa golurile)

RUN — șterge toate variabilele și începe rularea programului; indicarea unui număr de linie determină începerea programului de la acea linie

SAVE — memorarea unui program Basic prin perforare pe bandă; SAVE P salvează programul Basic, existent sub numele P; numele poate fi constituit doar dintr-o literă

### Comanda EDIT

Comanda EDIT permite corectarea unei linii de program; EDIT 10 corectează linia 10, dacă există erori.

Asociat, se mai dau și alte comenzi, formate din litere și cifre, care nu vor fi transmise la consolă. Numerele notate cu  $n$  sînt în gama 1—255.

A — încarcă din nou tamponul EDIT din memoria program

$nD$  —  $n$  numere vor fi șterse

E — încheie EDIT și înlocuiește linia

$nFx$  — găsește al  $n$ -lea caracter  $x$  în tampon și oprește indicatorul (pointer) în fața lui

H — șterge tot ce este în dreapta pointer-ului și intră în modul INSERT

I — inserează toate caracterele următoare, pînă cînd se introduce CR sau ESC

$nKx$  — șterge toate caracterele de la pointer pînă la al  $n$ -lea caracter  $x$  care nu este însă șters

L — editarea liniilor (listare)

Q — părăsește EDIT fără înlocuire (QUIT)

$nR$  — înlocuiește următoarele  $n$  caractere prin cele  $n$  precizate (replace)

X — pointer-ul la sfîrșitul liniilor și intră în modul INSERT

Space — pointer-ul la dreapta

Rubout — pointer-ul la stînga

CR (↵; retur de car) — încheie EDIT, cu înlocuire

Escape — încheie modul INSERT

Comenzi pentru consolă

LIST — editarea unui program; LIST 10—100 se referă la liniile de la 10 pînă la 100; LIST 20 se referă la toate liniile începînd cu 20

LVAR — editarea tuturor variabilelor actuale

NULL — pentru console lente, NULL 3,255 face ca după fiecare CR LF să se trimită 3 caractere ASCII FF<sub>h</sub> (Rubout)

POS — se dă poziția actuală a locului de imprimare la consolă; de exemplu, în A=POS(B), unde B este o valoare numerică.

PRINT — comandă de editare, de exemplu: PRINT 123,A, „TEST”,B,CS; prin introducerea unei virgule vor fi poziționate toate cele 14 coloane; prin punct și virgulă se determină lăsarea a 2 spații libere; dacă se încheie cu virgulă sau punct și virgulă, nu va fi trimis la ieșire ansamblul de caractere CR LF

PRINT USING — are două forme: PRINT USING linie; listă  
PRINT USING șir; listă



În al doilea exemplu formatul va fi determinat de variabilele șirului; la introducerea unei linii, trebuie început cu „I” pentru ca formatul să fie respectat

- # — zonă numerică
- . — poziție zecimală
- + — poate sta la începutul unei expresii aritmetice
- — extindere a semnului +; numerele pozitive vor fi scrise la consolă cu un spațiu înainte
- \*\* — spațiile vide vor fi completate cu \*
- SS — S va fi dat pentru prima cifră
- \*\*S — combinație de două cifre
- , — o virgulă în stînga punctului zecimal indică legarea tuturor celor trei poziții
- ↑↑↑↑ — patru caractere de acest tip arată că numărul este scris în formă exponențială

Șirurile sînt însoțite de "" ; pot fi urmate de unul sau mai multe dintre caracterele :

- L — egalitate la stînga
- R — egalitate la dreapta
- C — centrare
- E — egalare la stînga cu extindere în cazul în care șirul este prea lung
- SPC — permite emiterea unui număr de spații vide, de exemplu în PRINT A; SPC (5); B, vor fi 5 spații vide între A și B
- SWITCH — permite schimbarea configurației la consolă; comanda conține un argument între 0 și 3: 0=TTY, 1=CRT, 2=BATCH USE, 3=USER DEFINED
- TAB — se poate atinge o anumită poziție de imprimare: PRINT A; TAB (30); B determină imprimarea lui B începînd cu poziția 30
- TRACE — TRACE 1 introduce modul Trace, iar TRACE 0 determină ieșirea din acest mod; în locul numărului se poate afla și o expresie aritmetică; dacă este diferită de 0, atunci face legătura; se emit toate numerele de linii efectuate în "< >"

WIDTH — permite fixarea lungimii unei linii, cu introducerea caracterelor CR LF; WIDTH 80 stabilește lungimea de 80 de caractere pe linie; minimul este 15, iar maximul 255.

### Comenzi pentru imprimanta de linie (LIST — Device)

Cele mai multe dintre comenzile de mai sus pot fi utilizate și cu o imprimantă, utilizînd următoarele variante: LLIST, LLVAR, LNULL, LPRINT, LPRINT USING, LTRACE, LWIDTH, LPOS, SPC, TAB.

### Comenzi și funcții pentru transferul datelor

LET — atribuie o valoare unei variabile, aflată la stînga semnului „=”; de exemplu 10 LET A=20; LET poate fi și omisă

DIM — rezervare de spațiu de memorie pentru matrice (pot avea dimensiuni între 1 și 255); de exemplu: 200 DIM A(10), B(40); 210 DIM C(50,10); 220 DIM D(J); 230 DIM AS(221)

DATA — depozitare de constante, care pot fi introduse prin READ; de exemplu: 10 DATA 5, 8, 7, 9, 1, 4

READ — introducere de constante, depozitate cu DATA; de exemplu: 20 READ A prima dată, A ia valoarea 5 iar în continuare 8, 7, 9, 1 și, în sfîrșit, 4.

RESTORE — permite poziționarea la început al indicatorului de citire pentru READ; prin indicarea unui număr de linie, indicatorul poate fi poziționat pe o anumită linie, de exemplu 200 RESTORE 30

LINE INPUT — se poate citi o linie întreagă într-un șir de variabile; forma generală este LINE INPUT „prompt string”; lista de intrare este opțională



INPUT — permite intrarea datelor prin consolă; de exemplu:  
 10 INPUT „DATI A,B,C”; A,B,C  
 20 INPUT BS  
 INP — citirea unui port Z80; de exemplu pentru  $A = \text{INP}(0)$ , A va conține valoarea din canalul 0  
 OUT — de exemplu OUT 1,7 dă valoarea 7 canalului 1  
 WAIT — buclă de așteptare automată pentru porturi: WAIT A,B,C determină ca valoarea de la portul A să fie combinată printr-un SAU exclusiv cu cea de la portul C și printr-un SI cu cea de la portul B; când rezultatul este nenul, programul Basic va fi continuat  
 PEEK — permite accesul la memoria fizică;  $B = \text{PEEK}(A)$  aduce conținutul zecimal al celulei de memorie cu adresa zecimală A, în B  
 POKE — permite scrierea în memoria fizică; POKE A,B determină înscrierea lui B în locația cu numărul A  
 COPY — permite deplasarea sau dublarea unor porțiuni din programul Basic; formatul este COPY X, Y, Z unde X este adresa de început, Y este adresa de sfârșit și Z este numărul de linii de copiat  
 COPY linii noi, increment=domeniu de linii  
 și are ca efect copierea liniilor din domeniul de linii în noul domeniu și renumerotarea lor  
 EXCHANGE — permite schimbarea rapidă de valori de variabile; de exemplu EXCHANGE A\$, B\$ sau EXCHANGE C,D(I,J); în cazul șirurilor, se vor schimba intern doar indicatorii

### Ramificarea programelor

GOTO — instrucțiune de salt (argumentul=număr de linie)  
 GOSUB — apel de subprogram (argumentul=număr de linie)  
 RETURN — revenire din subprogram  
 ON × GOTO, ON × GOSUB — se va efectua saltul sau apelul de la numărul de linie n; de exemplu în 10 ON A GOTO 100, 125, 145 se efectuează salt la 100 dacă  $A=1$ ; dacă  $A=0$  sau mai mare decât numerele de linie date, se va executa următoarea linie  
 CALL — apel de subprogram — mașină; CALL adresă, argument 1, ..., argument n apelează un subprogram cu adresa indicată, fiecare argument fiind transformat într-un număr de 16 biți, transmis la subrutină după următoarea schemă:  
 SP — argumentul n  
 .....  
 argumentul 1  
 HL — adresa de revenire  
 BC — numărul argumentelor de pe stivă  
 FOR, TO, STEP, NEXT — instrucțiuni de ramificare, care pot fi utilizate în diferite moduri; de exemplu:  
 10 FOR A=B TO C STEP D  
 20 ...  
 30 NEXT A  
 IF, THEN, ELSE — instrucțiuni pentru a decide ramificarea programului în urma testării unei condiții; de exemplu:  
 10 IF B=4 THEN 50 ELSE 30  
 20 IF Z\$="DA" GOTO 50  
 30 IF C=5 THEN C=4 ELSE C=7  
 Ca și operatori de comparare sunt admisi: = (egal), <> (neegal, diferit de), < (mai mic decât), > (mai mare decât), <= (mai mic sau egal cu), >= (mai mare sau egal cu); operatorii logici admisi sunt NOT (negație), AND (SI logic) și OR (SAU logic); de exemplu:  
 20 IF (A=0) OR NOT (B=4) THEN C=5



## Funcții trigonometrice

ATN — arctangentă:  $A = \text{ATN}(.45)$ ; rezultatul se obține în radiani  
COS — cosinus, cu unghiul dat în radiani; de exemplu  $B = \text{COS}(3.141)$   
SIN — sinus, de exemplu  $C = \text{SIN}(3.1415/2)$   
TAN — tangentă, de exemplu  $A = \text{TAN}(.254)$

## Funcții diverse

ABS — valoarea absolută; de exemplu  $\text{ABS}(-4.5)$  are rezultatul 4.5

DEF FN — servește pentru definirea de către utilizator a unor funcții proprii; o funcție de acest tip trebuie să înceapă cu FN, urmat de un nume de variabilă, de exemplu FNA, FNB6; numele funcției este urmat de un parametru aflat într-o paranteză; de exemplu în

```
200 DEF FNQ(X)=X*B+3
```

B este o variabilă considerată globală în programul Basic, iar X este o variabilă locală, limitată la definiție și reprezintă numai parametrul; în exemplul următor, rezultatul va fi 9:

```
10 DEF FNA(X)=X*X
```

```
100 A=FNA(3)
```

```
110 PRINT A
```

Este posibilă și extinderea instrucțiunii DEF peste mai multe linii; pot fi definite și funcții recursive, formatul general fiind:

```
DEF FN nume (parametru, ..., parametru)
```

Corpul funcției

```
FNEND valoarea funcției
```

Spre deosebire de definiția standard, aici se poate omite semnul de egalitate; definirea funcției este încheiată prin FNEND, unde se indică și valoarea ei; funcția poate fi întreruptă și mai repede, revenirea fiind posibilă cu „FNRETURN valoarea funcției”, ca în exemplul următor:

```
100 DEF FNFAC(I)
```

```
200 IF I=0 THEN FNRETURN I
```

```
300 FNEND FNFAC (I-1)*I
```

```
400 PRINT FNFAC(6)
```

Alt exemplu:

```
100 DEF FNREPS(IS,I)'Construirea unui șir repetabil'
```

```
200 JS=""
```

```
300 IF I=0 THEN FNRETRUN JS
```

```
400 FOR J=1 TO I
```

```
500 JS=JS+1
```

```
600 NEXT
```

```
700 FNEND JS
```

```
800 PRINT FNREPS ("TEST",5)
```

EXP — funcție exponențială; de exemplu  $\text{EXP}(1)$  are rezultatul 2,7182...

FRE — furnizează utilizatorului memoria liberă pentru variabile și program; dacă parametrul este o variabilă, se dă memoria disponibilă pentru variabile iar dacă se dă o variabilă șir, se furnizează zona liberă șir; de exemplu

FRE(X) furnizează zona liberă pentru variabile

FRE(XS) — furnizează zona liberă de memorie pentru șiruri

INT — partea întreagă a unui număr; în  $C = \text{INT}(A)$ , rezultatul va fi  $C=4$ , dacă  $A=4,56$



LOG — logaritmul natural al unui număr; LOG(EXP(1)) are ca rezultat valoarea 1

SGN — semnul unui număr real  $x$  (+1 dacă  $x > 0$ , 0 dacă  $x = 0$ , -1 dacă  $x < 0$ ); de exemplu SGN(56.5) are rezultatul +1

SQR — rădăcina pătrată a unui număr pozitiv; de exemplu SQR(2) are ca rezultat 1,4142

RND — producerea unui număr pseudo-aleator între 0 și 1; pentru aceasta RND necesită un parametru; dacă valoarea este mai mică decât 0, secvența RND este inițializată; dacă argumentul este 0, se transmite valoarea anterioară; un argument mai mare decât 0 dă valoarea următoare, secvențial; de exemplu, în  $R = \text{RND}(I)$ , R este un număr între 0 și 1.

RANDOMIZE — permite fixarea unui punct de start arbitrar într-o serie de numere pseudo-aleatoare

### Prelucrarea șirurilor de caractere

ASC — furnizează valoarea zecimală a primului caracter dintr-un șir; de exemplu  $A = \text{ASC}(A\$)$ , cu  $A\$ = "A"$ , atribuie lui A valoarea 65 ( $65_D = 41_H$ )

CHR\$ — determină caracterul care corespunde unei valori zecimale a argumentului; codificarea se face în ASCII; de exemplu în  $A\$ = \text{CHR}\$(66)$ , A\$ va conține caracterul B ( $66_D = 42_H$ , care este codul ASCII al caracterului B)

LEFT\$ — conține 2 parametri; primul parametru este un șir; al doilea parametru dă numărul caracterelor care trebuie furnizate de la marginea stângă a șirului; de exemplu

30 BS = LEFT\$("STRING", 2)  
în care BS are semnificația "ST"

LEN — permite determinarea lungimii unui șir; de exemplu în  $X = \text{LEN}(S\$)$ , X reprezintă numărul de octeți conținuți în S\$

MID\$ — necesită 3 parametri: primul precizează șirul, al doilea desemnează poziția de start, iar al treilea numărul de caractere care vor fi utilizate în continuare; de exemplu dacă  $A\$ = \text{MID}\$(B\$, 5, 6)$ , A\$ conține 6 caractere, începând cu al 5-lea caracter al șirului B\$;

RIGHT\$ — funcționează ca și LEFT\$, dar aici caracterele sînt considerate începînd de la marginea din dreapta a șirului

STR\$ — precizează șirul a cărui valoare numerică se dă între paranteze; de exemplu în,  $C\$ = \text{STR}\$(7.8)$ , C\$ conține șirul "7.8"

VAL — opusul lui STR\$; se atribuie valoarea numerică a unui șir; în exemplul  $A = \text{VAL}("3.4")$ , A primește valoarea 3.4; în programele Basic se pot utiliza și constante hexazecimale, precedate de caracterul \$; de exemplu,  $B = \$400$  determină valoarea  $1024_D (400_H)$  pentru B

INSTR — servește la căutarea unui șir; pentru aceasta se indică primul parametru al șirului, în care al doilea parametru, șirul căutat, trebuie să fie găsit; suplimentar pot fi indicate și poziția de start și lungimea; de exemplu:

INSTR("123456789", "456") are ca rezultat valoarea 4

INSTR("123456789", "654") are ca rezultat valoarea 0

INSTR("1234512345", "34") are ca rezultat valoarea 3

INSTR("1234512345", "34", 6) are ca rezultat valoarea 8

INSTR("1234512345", "34", 6, 2) are ca rezultat valoarea 0

### Alte comenzi

END — termină programul; poate sta oriunde în programul Basic

REM — indică faptul că linia reprezintă un comentariu; poate fi utilizat în locul lui REM și caracterul "; de exemplu:

10 REM linie de comentariu



20 A=B' comentariu

STOP — ca și END, dar se emite și BREAK @ LINE...

USR — permite apelul unui subprogram-mașină; programul este apelat prin adresa de salt.

Pentru a obține parametrul, subprogramul trebuie apelat la adresa  $300_H + 27_H$ . Adresa parametrului se păstrează în perechea de registre DE. Pentru a readuce informația, subprogramul trebuie apelat la adresa  $300_H + 2A_H$ . Octetul mai puțin semnificativ al rezultatului va fi livrat în registrul B, iar cel mai semnificativ, în registrul A. Pentru revenirea în sistemul Basic, se execută instrucțiunea RET.

Exemplu:

10 A=USR(B)

20 PRINT A

### Comenzi pentru intrarea/ieșirea ASCII a programelor

ASAVE — la executarea acestei comenzi este emis programul Basic existent în memorie, în cod ASCII prin canalul de perforare (TTY)

ALOAD, ALOADC, AMERGE, AMERGE C — servesc pentru încărcarea programelor în cod ASCII (deci în formă de cod obiect); fiecare linie trebuie să înceapă cu un număr de linie și trebuie să se termine cu un caracter CR; citirea este terminată prin CTRL-Z în textul de intrare sau prin EOF în subprogramul de citire; ALOAD șterge un program anterior, MERGE mixează liniile următoare cu programul existent; deosebirea dintre comenzile "A..." și "A...C" se află în modul de tratare al cititorului; "A...C" presupune un cititor care poate fi comandat ca după intrarea fiecărei linii, aceasta să fie transformată în formatul intern; "A..." permit citirea pînă la capăt a programului, după care urmează conversia în formatul intern; în acest caz este necesar un spațiu mai larg de memorie.

### Caractere de comandă

- COMMA — pentru poziția TAB următoare, sau caracter de separare.
- SEMI-COLON — a nu se depăși
- COLON — pentru mai multe indicații pe linie
- RUBOUT — ștergerea caracterului dat între "/"
- CTRL-S — oprirea ieșirii datelor
- CTRL-Q — continuarea ieșirii
- CTRL-C — terminarea executării programului Basic
- CTRL-U — ștergerea liniei indicate
- CTRL-X — revenirea în programul monitor
- CTRL-O — suprimarea ieșirii la consolă
- CTRL-R — ieșirea liniei actuale fără caracter de ștergere
- CTRL-T — în timpul rulării programului poate fi dat numărul de linii deja prelucrate

Operatorii utilizați, în ordinea prelucrării lor (ierarhic), sînt :

- ( ) paranteză
- ↑ operator exponențial
- negație
- \* multiplicare
- / diviziune
- + adunare
- scădere
- = egalitate
- < > inegalitate
- < mai mic decît



/ mai mare decât  
 / = mai mic/egal  
 / = mai mare/egal  
 NOT NU logic  
 AND SI logic  
 OR SAU logic

Listarea interpretorului Basic, inclusiv a salturilor care asigură legătura cu sistemul de operare și care se află în zona de memorie RAM 0000<sub>H</sub>—02FF<sub>H</sub>, este următoarea:

0000	E5	D5	C5	E5	D5	C5	1E	80	CD	B3	E6	7B	C1	D1	E1	E6
0010	82	CA	18	02	AF	C9	3E	FF	C9	D1	40	B9	60	21	60	3D
0020	3E	00	06	3F	C9	09	6F	35	69	3C	60	70	6B	65	6A	46
0030	E5	D5	C5	1E	02	CD	B3	E6	C1	D1	E1	C9	83	3B	6F	E2
0040	E5	D5	C5	1E	02	51	CD	80	E6	C1	D1	E1	C9	D6	84	12
0050	76	52	75	4B	67	AB	51	35	50	61	56	5B	19	E8	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0100	BF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0110	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0120	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0130	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0140	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0150	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0160	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0170	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0180	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0190	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
01A0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
01B0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
01C0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
01D0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
01E0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
02F0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0200	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0210	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0220	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0230	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0240	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0250	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0260	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0270	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0280	DF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0290	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
02A0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
02B0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00



02C0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
02D0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
02E0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
02F0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
0300	C3	C9	2F	C3	73	04	C3	44	0A	C3	30	00	C3	1D	E1	C3
0310	40	00	C3	1D	E1	C3	40	00	C3	03	00	C3	14	00	C3	15
0320	00	C3	20	00	C3	1D	E1	C3	2D	0A	C3	1E	12	A3	22	AD
0330	23	CE	22	06	03	F6	11	D4	16	2F	12	0E	2A	0E	2B	EB
0340	20	64	2A	4A	2B	50	2B	D3	2B	EA	2B	6A	1F	99	15	BA
0350	13	2B	17	A8	15	B6	15	C6	15	F6	15	FF	15	2A	12	2E
0360	16	79	A7	24	79	D8	1F	7C	3B	21	7C	BF	21	7F	17	2A
0370	50	BD	0F	46	BC	0F	C3	09	6A	08	3B	0E	0B	0B	55	0D
0380	65	10	A6	0D	27	0B	C5	0A	1D	13	23	0C	6A	09	A8	0A
0390	E6	0A	0D	0B	C1	09	DD	16	06	0C	FC	09	E3	16	36	12
03A0	71	1F	5C	0C	5C	0C	55	0D	72	0A	1D	13	C5	17	0D	0B
03B0	0D	0B	59	0C	DB	1A	D8	1A	84	1C	46	17	62	17	F9	09
03C0	65	17	9E	1C	9B	1C	5C	0C	6D	04	59	1F	26	1F	F9	1E
03D0	B6	1E	4F	0D	B0	1D	9D	0A	FE	17	B3	05	24	0D	F2	1D
03E0	6B	1D	6E	1D	F2	1C	F5	1C	41	18	EF	07	EC	07	B3	18
03F0	A3	1A	FE	1A	E7	09	21	04	00	39	01	11	00	7E	23	FE

0400	81	C0	7E	23	E5	66	6F	7A	B3	EB	28	07	EB	7C	92	20
0410	02	7D	93	E1	23	C8	09	18	E4	CD	3E	04	C5	E3	C1	B7
0420	E5	ED	52	C5	E3	C1	EB	E3	03	ED	B8	23	13	42	4B	D1
0430	C9	E5	2A	62	01	06	00	09	09	CD	3E	04	E1	C9	D5	EB
0440	21	D8	FF	39	7C	92	20	02	7D	93	EB	D1	D0	1E	07	18
0450	2D	BE	CA	47	09	18	16	3A	4F	01	B7	20	0A	C1	21	84
0460	2F	CD	BD	07	C3	F4	05	2A	4C	01	22	54	01	1E	02	01
0470	1E	0B	01	1E	16	01	1E	0A	01	1E	12	01	1E	01	CD	DB
0480	05	CD	F8	05	CD	EE	0C	21	12	2E	1D	28	07	CB	7E	23
0490	28	FB	18	F6	CD	BD	07	2A	54	01	7C	A5	3C	C4	C1	24
04A0	CD	F8	05	32	AB	01	CD	DB	0C	21	FF	FF	22	54	01	3A
04B0	AB	01	B7	28	25	2A	9A	01	FA	D5	1D	ED	5B	9C	01	19
04C0	DA	6B	11	22	9A	01	E5	CD	C9	24	CD	64	07	CD	07	07
04D0	CD	47	09	D1	B7	28	C9	37	18	0A	CD	07	07	CD	47	09
04E0	3C	3D	28	C5	F5	FE	03	CA	90	09	D9	21	AB	01	34	35
04F0	D9	CC	49	0A	FE	20	20	01	23	D5	CD	09	06	47	D1	F1
0500	D2	14	09	D5	C5	23	7E	B7	28	0C	FE	20	28	F7	FE	09
0510	28	F3	FE	0A	28	EF	F5	CD	92	05	C5	30	14	EB	2A	5E
0520	01	1A	02	03	13	7C	92	20	02	7D	93	20	F4	ED	43	5E
0530	01	D1	F1	28	21	ED	4B	5E	01	E1	09	E5	CD	19	04	E1
0540	22	5E	01	EB	74	23	23	D1	73	23	72	23	11	B1	01	1A
0550	77	23	13	B7	20	F9	CD	C2	05	23	EB	21	A9	04	E5	62
0560	6B	7E	23	B6	C8	23	23	23	AF	BE	23	20	FC	EB	73	23
0570	72	18	EC	11	00	00	D5	28	0C	D1	CD	85	09	D5	28	0E
0580	3E	A6	CD	51	04	11	FF	FF	C4	85	09	C2	6D	04	EB	D1
0590	E3	E5	2A	5C	01	44	4D	7E	23	B6	2B	C8	23	23	7E	23
05A0	66	6F	ED	52	60	69	7E	23	66	6F	3F	C8	3F	D0	D9	19
05B0	D9	18	E2	C0	2A	5C	01	AF	32	A3	01	77	23	77	23	22
05C0	5E	01	2A	5C	01	2B	22	50	01	2A	04	01	22	48	01	CD
05D0	7B	09	2A	5E	01	22	60	01	22	62	01	C1	ED	7B	5A	01
05E0	21	08	01	22	06	01	AF	67	6F	22	58	01	32	4E	01	32
05F0	A7	01	E5	C5	2A	50	01	C9	AF	32	00	01	FD	21	0F	03



0600	FD	22	8E	01	FD	21	90	01	C9	AF	32	03	01	0E	05	11
0610	B1	01	7E	47	B7	CA	AD	06	FE	21	38	4F	FE	22	28	7B
0620	3A	03	01	B7	47	7E	20	43	FE	30	38	04	FE	3A	38	3B
0630	D5	11	63	2C	E5	18	02	23	13	1A	B7	28	25	FE	20	20
0640	09	13	7E	FE	20	20	03	23	18	F8	EB	1A	FE	60	38	02
0650	E6	5F	AE	E6	7F	EB	20	49	1A	17	30	DB	F1	78	F6	80
0660	18	08	E1	7E	FE	60	38	02	E6	5F	D1	EB	FE	C9	36	3A
0670	20	02	0C	23	EB	23	12	13	0C	D6	3A	28	0A	FE	9B	06
0680	3A	28	11	FE	49	20	03	32	03	01	D6	54	28	05	D6	0E
0690	C2	12	06	47	7E	B7	28	15	B8	28	D0	23	12	0C	13	18
06A0	F3	E1	E5	04	EB	CB	7E	23	28	FB	EB	18	8C	21	B0	01
06B0	12	13	12	13	12	C9	78	3D	FD	B6	00	28	4A	CD	70	07
06C0	10	08	CD	70	07	CD	70	07	18	3A	2B	7E	CD	66	07	CD
06D0	C9	07	FE	7F	28	EA	F5	CD	70	07	F1	18	36	3E	3F	CD
06E0	72	07	3E	3F	CD	72	07	CD	64	07	CD	07	07	23	7E	2B
06F0	FE	03	C9	CD	FC	05	F5	3E	5E	CD	72	07	F1	F6	40	18
0700	71	CD	F6	06	CD	0E	0D	21	B1	01	06	01	78	32	A7	01
0710	CD	C9	07	FE	07	28	3C	FE	0D	CA	E9	0C	FE	03	CA	E1
0720	0C	FE	15	28	DC	FE	7F	28	8D	FE	12	20	14	CD	F6	06
0730	CD	0E	0D	21	B1	01	48	0D	28	D6	7E	23	CD	66	07	18
0740	F6	FE	09	28	0E	FE	0A	20	06	05	28	BB	04	18	04	FE
0750	20	38	BD	4F	78	FE	FD	3E	07	30	04	79	77	23	04	CD
0760	66	07	18	AC	3E	20	FE	0A	20	08	CD	EE	0C	3E	0A	C9
0770	3E	5C	C5	4F	3A	00	01	B7	20	3F	79	FE	09	20	27	FD
0780	7E	01	E6	F8	3D	FD	BE	00	38	17	FD	7E	00	E6	07	2F
0790	C6	09	47	0E	20	CD	8D	01	FD	34	00	10	F8	0E	09	18
07A0	18	CD	EE	0C	18	F7	FE	20	38	0C	FD	7E	00	FD	BE	01
07B0	CC	EE	0C	FD	34	00	CD	8D	01	79	C1	B7	C9	7E	CB	BF
07C0	CD	66	07	CB	7E	C0	23	18	F4	CD	09	03	E6	7F	FE	18
07D0	28	0F	FE	0F	C0	CD	F3	06	3A	00	01	2F	32	00	01	AF
07E0	C9	CD	F3	06	CD	EE	0C	CD	24	03	AF	C9	CD	A6	18	C1
07F0	CD	73	05	C5	CD	EE	0C	E1	D1	4E	23	46	23	78	B1	28

0800	33	CD	93	09	C5	4E	23	46	23	C5	E3	EB	7C	92	20	02
0810	7D	93	C1	38	1E	E3	E5	C5	EB	CD	C9	24	E1	CD	64	07
0820	CD	37	08	21	B1	01	01	F4	07	C5	2B	06	00	CD	F0	13
0830	C3	34	14	C1	C3	A0	04	01	B0	01	3E	E1	7E	03	B7	23
0840	02	C8	F2	3C	08	FE	C9	20	01	0B	D6	80	E5	21	63	2C
0850	28	08	CB	7E	23	28	FB	3D	20	F8	7E	FE	20	28	08	B7
0860	CB	BF	02	FA	3B	08	03	23	18	F0	3E	29	32	4E	01	CD
0870	27	0B	E3	CD	F6	03	D1	20	02	09	F9	EB	0E	0A	CD	31
0880	04	E5	CD	0B	0B	E3	E5	2A	54	01	E3	3E	9F	CD	51	04
0890	CD	88	0E	CD	85	0E	E5	CD	02	23	E1	C5	DD	E5	D5	01
08A0	00	81	DD	21	00	00	51	59	7E	FE	A4	3E	01	20	0E	CD
08B0	47	09	CD	85	0E	E5	CD	02	23	CD	BF	22	E1	C5	DD	E5
08C0	D5	F5	33	E5	2A	50	01	E3	06	81	C5	33	CD	18	03	3C
08D0	CC	98	09	22	50	01	7E	FE	3A	28	39	B7	C2	6D	04	23
08E0	7E	23	E6	23	CA	C8	09	5E	23	56	ED	53	54	01	3A	A3
08F0	01	B7	28	20	F5	FC	A6	18	D5	E5	3E	3C	CD	72	07	EB
0900	CD	C9	24	3E	3E	CD	72	07	E1	D1	F1	87	30	06	CD	FC
0910	05	87	38	E0	CD	47	09	11	CC	08	D5	C8	D6	80	DA	2B
0920	0B	FE	46	CA	8B	0B	FE	1D	38	0F	D6	49	DA	6D	04	FE
0930	23	D2	6D	04	11	B0	03	18	03	11	76	03	07	4F	06	00
0940	EB	09	4E	23	46	C5	EB	23	7E	FE	D5	20	08	23	7E	FE



0950	3A	C8	B7	20	F8	FE	3A	D0	FE	20	28	EB	FE	09	28	E7
0960	FE	0A	28	E3	FE	30	3F	3C	3D	C9	28	0F	CD	85	09	E5
0970	CD	92	05	D1	D2	E1	0A	60	69	18	04	EB	2A	5C	01	2B
0980	22	64	01	EB	C9	2B	CD	47	09	CD	49	0A	18	BA	C1	C1
0990	B7	18	34	CD	18	03	3C	C0	CD	C9	07	FE	14	20	0C	E5
09A0	2A	54	01	7C	A5	3C	C4	9D	1A	E1	C9	FE	13	20	0B	CD
09B0	C9	07	FE	03	28	04	FE	11	20	F5	FE	03	C0	CD	F3	06
09C0	3E	C0	F6	C0	22	50	01	C1	F5	2A	54	01	7D	A4	3C	28
09D0	09	22	56	01	2A	50	01	22	58	01	CD	F8	05	F1	21	AF
09E0	2F	C2	94	04	C3	A0	04	1E	11	2A	58	01	7C	B5	28	56
09F0	EB	2A	56	01	22	54	01	EB	C9	CD	9F	18	CD	1B	17	FE
0A00	32	30	41	FD	77	03	3E	2C	BE	C0	CD	47	09	CD	1B	17
0A10	FD	77	04	C9	7E	FE	41	D8	FE	5B	3F	C9	CD	88	0E	18
0A20	0C	CD	47	09	CD	85	0E	CD	BF	22	FA	44	0A	3A	6B	01
0A30	FE	91	DA	75	23	01	80	91	DD	21	00	00	11	00	00	CD
0A40	36	23	51	C8	1E	05	C3	7E	04	2B	11	00	00	23	7E	B7
0A50	C8	FE	30	3F	D0	FE	3A	D0	E5	21	98	19	B7	ED	52	DA
0A60	6D	04	62	6B	19	29	19	29	D6	30	5F	16	00	19	EB	E1
0A70	18	DB	28	26	CD	24	0A	CD	48	09	C0	E5	2A	04	01	ED
0A80	52	EB	DA	6D	04	2A	5E	01	01	28	00	09	7C	92	20	02
0A90	7D	93	D2	4D	04	EB	22	5A	01	E1	C3	C6	05	CA	C2	05
0AA0	CD	C6	05	01	CC	08	18	1C	0E	03	CD	31	04	C1	E5	ED
0AB0	5B	54	01	D5	7A	B3	3C	3A	A7	01	57	20	04	AF	32	A7
0AC0	01	1E	8C	D5	C5	CD	49	0A	CD	0D	0B	E5	2A	54	01	7C
0AD0	92	20	02	7D	93	E1	23	DC	95	05	D4	92	05	60	69	2B
0AE0	D8	1E	08	C3	7E	04	C0	16	FF	CD	F6	03	56	23	F9	FE
0AF0	8C	1E	03	20	EE	E1	22	54	01	7C	A5	3C	21	A7	01	7E
0B00	72	20	04	B7	C2	A5	13	21	CC	08	E3	01	3A	0E	00	06
0B10	00	79	48	47	7E	B7	C8	B8	C8	23	FE	22	28	F3	D6	8A
0B20	20	F2	B8	8A	57	18	ED	FE	C6	28	60	CD	6A	10	3E	AD
0B30	CD	51	04	D5	3A	02	01	F5	CD	9A	0E	F1	E3	22	50	01
0B40	1F	CD	8A	0E	20	07	E5	CD	16	23	D1	E1	C9	E5	2A	66
0B50	01	E5	23	23	5E	23	56	21	B0	02	7C	92	20	02	7D	93
0B60	30	18	2A	5A	01	7C	92	20	02	7D	93	D1	30	14	2A	5E
0B70	01	7C	92	20	02	7D	93	30	09	3E	D1	CD	83	15	EB	CD
0B80	CA	13	CD	83	15	E1	CD	19	23	E1	C9	CD	47	09	3E	28
0B90	CD	51	04	CD	6A	10	CD	89	0E	3E	2C	CD	51	04	D5	CD
0BA0	1B	17	B7	CA	44	0A	F5	1E	FF	7E	FE	29	28	08	3E	2C
0BB0	CD	51	04	CD	1B	17	3E	29	CD	51	04	3E	AD	CD	51	04
0BC0	F1	E3	3D	BE	06	00	30	08	4F	7E	91	BB	47	38	01	43
0BD0	C5	23	23	46	23	66	68	06	00	09	C1	E3	C5	CD	95	0E
0BE0	C1	D1	E5	2A	66	01	78	96	F5	78	38	01	7E	23	23	4E
0BF0	23	46	CD	59	15	F1	38	09	28	07	EB	36	20	23	3D	20

0C00	FA	CD	63	15	E1	C9	CD	1B	17	7E	47	FE	8C	28	06	3E
0C10	88	CD	51	04	2B	4B	0D	78	CA	1C	09	CD	86	09	FE	2C
0C20	C0	18	F3	CD	9A	0E	7E	FE	2C	CC	47	09	FE	88	28	06
0C30	3E	A2	CD	51	04	2B	E5	CD	BF	22	E1	28	09	CD	47	09
0C40	DA	C5	0A	C3	1B	09	16	01	CD	0B	0B	B7	C8	CD	47	09
0C50	FE	C9	20	F4	15	20	F1	18	E4	CD	9F	18	CA	EE	0C	FE
0C60	9D	CA	21	28	FE	9E	28	4E	FE	A1	28	49	E5	FE	2C	28
0C70	32	FE	3B	28	5F	C1	CD	9A	0E	E5	3A	02	01	B7	20	1A
0C80	CD	DD	24	CD	ED	13	2A	66	01	FD	7E	00	86	FD	BE	01
0C90	D4	EE	0C	CD	34	14	CD	64	07	AF	C4	34	14	E1	CD	48



0CA0	09	18	B9	FD	7E	00	FD	BE	02	D4	EE	0C	30	26	D6	0E
0CB0	30	FC	2F	18	18	37	F5	CD	18	17	3E	29	CD	51	04	2B
0CC0	F1	E5	3E	FF	38	04	FD	7E	00	2F	83	30	07	3C	47	CD
0CD0	64	07	10	FB	E1	CD	47	09	C8	18	81	21	9B	2F	C3	A4
0CE0	01	21	B1	01	77	23	CD	F6	06	36	00	21	B0	01	3E	0D
0CF0	FD	77	00	CD	72	07	3E	0A	CD	72	07	FD	7E	03	3C	3D
0D00	FD	77	00	C8	F5	FD	7E	04	CD	72	07	F1	18	F1	CD	EE
0D10	0C	3A	AB	01	B7	C8	E5	C5	2A	9A	01	CD	C9	24	CD	64
0D20	07	C1	E1	C9	D1	11	0A	00	D5	DC	85	09	EB	E3	EB	FE
0D30	2C	20	06	CD	47	09	DC	85	09	B7	C2	6D	04	ED	53	9C
0D40	01	7A	B3	CA	6D	04	3E	01	32	AB	01	E1	C3	C3	04	3E
0D50	84	CD	51	04	F6	AF	F5	7E	FE	22	3E	00	32	00	01	20
0D60	0D	CD	EE	13	3E	3B	CD	51	04	E5	CD	34	14	E1	E3	E5
0D70	CD	9C	13	CD	E2	06	CA	8E	09	F1	28	2F	E3	CD	6A	10
0D80	CD	89	0E	E3	D5	06	00	CD	F0	13	EB	21	93	0D	E3	D5
0D90	C3	4D	0B	E1	CD	48	09	C8	3E	2C	CD	51	04	E5	CD	DD
0DA0	06	CA	8F	09	18	D6	E5	2A	64	01	F6	AF	32	4F	01	E3
0DB0	18	05	3E	2C	CD	51	04	CD	6A	10	E3	D5	7E	FE	2C	28
0DC0	CC	3A	4F	01	B7	20	54	CD	DD	06	CA	8E	09	3A	02	01
0DD0	B7	28	1A	CD	47	09	57	47	FE	22	28	05	16	3A	06	2C
0DE0	2B	CD	F1	13	EB	21	F8	0D	E3	D5	C3	4D	0B	CD	47	09
0DF0	CD	DE	23	E3	CD	16	23	E1	CD	48	09	28	05	FE	2C	C2
0E00	57	04	E3	CD	48	09	20	AA	D1	3A	4F	01	B7	EB	C2	80
0E10	09	B6	21	A3	2F	D5	C4	BD	07	E1	C9	CD	0B	0B	B7	20
0E20	10	23	7E	23	B6	1E	04	28	69	23	5E	23	56	ED	53	4C
0E30	01	CD	47	09	FE	83	20	E3	C3	CD	0D	11	00	00	C4	6A
0E40	10	22	50	01	CD	F6	03	C2	7C	04	F9	D5	7E	23	F5	D5
0E50	CD	EC	22	E3	E5	CD	83	1F	E1	CD	16	23	E1	CD	05	23
0E60	E5	CD	36	23	E1	C1	90	CD	6C	23	28	09	EB	22	54	01
0E70	69	60	C3	C8	08	F9	2A	50	01	7E	FE	2C	C2	CC	08	CD
0E80	47	09	CD	3E	0E	CD	9A	0E	F6	37	3A	02	01	8F	B7	E8
0E90	1E	0D	C3	7E	04	CD	9A	0E	18	EF	2B	16	00	D5	0E	01
0EA0	CD	31	04	CD	14	0F	22	52	01	2A	52	01	C1	78	FE	78
0EB0	D4	88	0E	7E	16	00	D6	AC	38	15	FE	03	30	11	FE	01
0EC0	17	AA	BA	57	DA	6D	04	22	4A	01	CD	47	09	18	E7	7A
0ED0	B7	C2	E2	0F	7E	22	4A	01	D6	A5	D8	FE	07	D0	5F	3A
0EE0	02	01	3D	B3	7B	CA	12	15	07	83	5F	21	61	03	19	78
0EF0	56	BA	D0	23	CD	88	0E	C5	01	A9	0E	C5	ED	4B	66	01
0F00	C5	ED	4B	68	01	C5	ED	4B	6A	01	C5	4E	23	46	C5	2A
0F10	4A	01	18	89	AF	32	02	01	CD	47	09	DA	E3	23	CD	14
0F20	0A	30	44	FE	A5	28	ED	FE	2E	CA	E3	23	FE	A6	28	26
0F30	FE	22	CA	EE	13	FE	A3	CA	45	10	FE	A0	CA	6C	12	FE
0F40	26	CA	71	24	D6	AF	30	30	3E	28	CD	51	04	CD	9A	0E
0F50	3E	29	CD	51	04	C9	16	7D	CD	9D	0E	2A	52	01	E5	CD
0F60	D2	22	CD	88	0E	E1	C9	CD	6A	10	E5	EB	22	66	01	3A
0F70	02	01	B7	CC	EC	22	E1	C9	06	00	07	4F	C5	CD	47	09
0F80	79	FE	29	38	26	FE	30	28	22	FE	32	28	1B	D2	44	0A
0F90	3E	28	CD	51	04	CD	95	0E	3E	2C	CD	51	04	EB	2A	66
0FA0	01	E3	E5	EB	CD	1B	17	EB	E3	18	08	CD	48	0F	E3	11
0FB0	62	0F	D5	01	2D	03	09	4E	23	66	69	E9	F6	AF	F5	CD
0FC0	1C	0A	F1	EB	C1	DD	E1	E3	EB	CD	F5	22	F5	CD	2D	0A
0FD0	F1	C1	79	21	1D	12	20	05	A3	4F	78	A2	E9	B3	4F	78
0FE0	B2	E9	21	F4	0F	3A	02	01	1F	7A	17	5F	16	64	78	BA
0FF0	D0	C3	F7	0E	F6	0F	79	B7	1F	C1	DD	E1	D1	F5	CD	8A



1000	0E	21	3B	10	E5	CA	36	23	AF	32	02	01	D5	CD	63	15
1010	D1	4E	23	46	23	C5	4E	23	46	C5	CD	67	15	CD	6C	23
1020	E1	E3	55	E1	7B	B2	C8	7A	D6	01	D8	AF	BB	3C	D0	15
1030	1D	0A	BE	23	03	28	ED	3F	C3	CA	22	3C	8F	C1	A0	C6
1040	FF	9F	C3	A6	22	16	5A	CD	9D	0E	CD	88	0E	CD	2D	0A
1050	7B	2F	4F	7A	2F	CD	1D	12	C1	C3	A9	0E	CD	48	09	C8
1060	3E	2C	CD	51	04	01	5C	10	C5	F6	AF	32	01	01	46	CD
1070	14	0A	DA	6D	04	AF	4F	32	02	01	CD	47	09	38	05	CD
1080	14	0A	38	0B	4F	CD	47	09	38	FB	CD	14	0A	30	F6	D6
1090	24	20	09	3C	32	02	01	CB	F9	CD	47	09	3A	4E	01	3D
10A0	CA	3C	11	86	D6	27	28	6C	AF	32	4E	01	E5	2A	60	01
10B0	EB	2A	5E	01	7C	92	20	02	7D	93	28	12	79	96	23	20
10C0	02	78	96	23	28	40	23	23	23	23	23	23	18	E6	E1	E3
10D0	D5	11	6A	0F	7C	92	20	02	7D	93	D1	28	2C	E3	E5	C5
10E0	01	08	00	2A	62	01	E5	09	C1	E5	CD	19	04	E1	22	62
10F0	01	60	69	22	60	01	2B	36	00	7C	92	20	02	7D	93	20
1100	F5	D1	73	23	72	23	EB	E1	C9	32	6B	01	21	FF	02	22
1110	66	01	E1	C9	E5	2A	01	01	E3	57	D5	C5	CD	21	0A	C1
1120	F1	EB	E3	E5	EB	3C	57	7E	FE	2C	28	EE	3E	29	CD	51
1130	04	22	52	01	E1	22	01	01	1E	00	D5	11	E5	F5	2A	60
1140	01	3E	19	ED	5B	62	01	7C	92	20	02	7D	93	28	21	7E
1150	B9	23	20	02	7E	B8	23	5E	23	56	23	20	E5	3A	01	01
1160	B7	C2	76	04	F1	CA	01	1F	96	28	61	1E	09	C3	7E	04
1170	11	06	00	F1	CA	19	1F	71	23	70	23	4F	CD	31	04	23
1180	23	22	4A	01	71	23	3A	01	01	17	79	01	0B	00	30	02
1190	C1	03	71	23	70	23	F5	E5	CD	C6	23	EB	E1	F1	3D	20
11A0	EA	F5	42	4B	EB	19	38	C3	CD	3E	04	22	62	01	2B	36
11B0	00	7C	92	20	02	7D	93	20	F5	03	57	2A	4A	01	5E	EB
11C0	29	09	EB	2B	2B	73	23	72	23	F1	38	26	47	4F	7E	23
11D0	16	E1	5E	23	56	23	E3	F5	7C	92	20	02	7D	93	30	8B
11E0	E5	CD	C6	23	D1	19	F1	3D	44	4D	20	E5	29	09	29	C1
11F0	09	EB	2A	52	01	C9	2A	62	01	EB	21	00	00	39	3A	02
1200	01	B7	28	0D	CD	63	15	CD	73	14	2A	5A	01	EB	2A	48
1210	01	AF	32	02	01	ED	52	EB	AF	06	98	18	0A	41	50	1E
1220	00	21	02	01	73	06	90	C3	AB	22	3A	95	01	18	03	3A
1230	90	01	47	AF	18	E8	CD	AA	13	CD	9C	13	EB	73	23	72
1240	EB	2B	CD	47	09	28	07	FE	AD	20	F7	C3	0B	0B	B7	20
1250	0F	23	7E	23	B6	CA	6D	04	23	5E	23	56	ED	53	54	01
1260	CD	47	09	28	E9	FE	89	28	E2	C3	60	12	CD	AA	13	3A
1270	02	01	B7	F5	22	52	01	EB	7E	23	66	6F	B4	CA	79	04
1280	7E	FE	28	20	71	CD	47	09	22	4A	01	18	05	3E	2C	CD
1290	51	04	0E	05	CD	31	04	3E	29	32	4E	01	CD	6A	10	EB
12A0	3A	02	01	B7	37	20	10	4E	23	46	C5	23	4E	23	46	C5
12B0	23	4E	23	46	C5	18	08	F5	D5	EB	CD	10	14	D1	F1	E5
12C0	F5	EB	7E	FE	29	20	C6	2A	52	01	3E	28	CD	51	04	E5
12D0	2A	4A	01	CD	6A	10	E3	CD	33	0B	7E	FE	29	28	0D	3E
12E0	2C	CD	51	04	E3	3E	2C	CD	51	04	18	E7	CD	47	09	E3
12F0	3E	29	CD	51	04	3E	D5	CD	48	09	28	10	3E	AD	CD	51
1300	04	CD	9A	0E	CD	48	09	C2	6D	04	18	3A	0E	02	CD	31
1310	04	ED	5B	54	01	D5	16	A0	D5	33	C3	CC	08	20	0C	CD
1320	0E	20	32	44	01	2F	32	02	01	18	09	CD	9A	0E	CD	48
1330	09	C2	6D	04	16	FF	CD	F6	03	F9	FE	A0	1E	17	C2	7E
1340	04	D1	ED	53	54	01	3A	02	01	3C	28	03	3D	20	2E	D1



1350	F1	30	13	20	37	E1	C1	70	2B	71	2B	C1	70	2B	71	2B
1360	C1	70	2B	71	18	EA	F5	D5	21	02	01	CB	7E	28	01	77
1370	B7	11	44	01	C4	10	14	E1	F1	1F	C3	8A	0E	ED	5B	66
1380	01	CD	83	15	21	44	01	CD	19	23	18	C3	CD	83	15	7E
1390	22	06	01	E1	77	23	23	71	23	70	18	B4	E5	2A	54	01
13A0	23	7C	B5	E1	C0	1E	0C	C3	7E	04	3E	A0	CD	51	04	F6
13B0	80	47	3E	29	32	4E	01	C3	6F	10	CD	88	0E	CD	DD	24
13C0	CD	ED	13	CD	63	15	01	C2	15	C5	7E	23	23	E5	CD	49
13D0	14	E1	4E	23	46	CD	E1	13	E5	CD	59	15	D1	C9	CD	49
13E0	14	21	44	01	E5	77	23	23	73	23	72	E1	C9	2B	06	22
13F0	50	E5	0E	FF	23	7E	0C	B7	28	06	BA	28	03	B8	20	F4

1400	FE	22	CC	47	09	E3	23	EB	79	CD	E1	13	11	44	01	3E
1410	D5	2A	06	01	22	66	01	3E	01	32	02	01	CD	19	23	7C
1420	92	20	02	7D	93	22	06	01	E1	7E	C0	1E	10	C3	7E	04
1430	23	CD	ED	13	CD	63	15	CD	6C	23	1C	1D	C8	0A	CD	66
1440	07	FE	0D	CC	FB	0C	03	18	F2	B7	0E	F1	F5	2A	5A	01
1450	EB	2A	48	01	4F	AF	47	ED	42	7C	92	20	02	7D	93	38
1460	07	22	48	01	23	EB	F1	C9	F1	1E	0E	28	C0	BF	F5	01
1470	4B	14	C5	2A	04	01	22	48	01	AF	08	ED	5B	5A	01	D9
1480	21	08	01	ED	5B	06	01	7C	92	20	02	7D	93	01	83	14
1490	20	3F	2A	5E	01	ED	5B	60	01	7C	92	20	02	7D	93	28
14A0	0A	CB	7E	23	23	CD	D2	14	18	EB	C1	EB	2A	62	01	ED
14B0	52	EB	28	44	CD	6C	23	CB	7B	E5	09	28	ED	E3	4E	06
14C0	00	09	09	23	D1	7C	92	20	02	7D	93	28	DE	D5	01	C4
14D0	14	C5	7E	23	23	5E	23	56	23	23	23	C8	B7	C8	D5	D9
14E0	C1	2A	48	01	ED	42	D9	D8	D9	62	6B	ED	42	D9	D0	D9
14F0	50	59	D9	08	E5	DD	E1	C9	08	D0	D9	2A	48	01	EB	4F
1500	06	00	09	2B	ED	B8	62	6B	13	DD	72	FD	DD	73	FC	C3
1510	76	14	C5	E5	2A	66	01	E3	CD	14	0F	E3	CD	89	0E	7E
1520	E5	2A	66	01	E5	86	1E	0F	DA	7E	04	CD	DE	13	D1	CD
1530	67	15	E3	CD	66	15	E5	2A	46	01	EB	CD	49	15	CD	49
1540	15	21	AC	0E	E3	E5	C3	0C	14	E1	E3	4E	23	46	23	7E
1550	23	66	6F	78	B1	C8	ED	B0	C9	60	69	4F	06	00	18	F3
1560	CD	89	0E	2A	66	01	EB	CD	83	15	EB	C0	D5	50	59	1B
1570	4E	2A	48	01	7C	92	20	02	7D	93	20	05	47	09	22	48
1580	01	E1	C9	2A	06	01	2B	2B	2B	46	2B	4E	2B	2B	7C	92
1590	20	02	7D	93	C0	22	06	01	C9	01	32	12	C5	CD	60	15
15A0	AF	57	32	02	01	7E	B7	C9	CD	9D	15	28	58	23	23	5E
15B0	23	56	1A	C3	32	12	3E	01	CD	DE	13	CD	1E	17	2A	46
15C0	01	73	C1	C3	0C	14	CD	FF	16	AF	E3	4F	3E	E5	E5	7E
15D0	B8	38	02	78	11	0E	00	C5	CD	49	14	C1	E1	E5	23	23
15E0	46	23	66	68	06	00	09	44	4D	CD	E1	13	CD	59	15	D1
15F0	CD	67	15	C3	0C	14	CD	FF	16	D1	D5	1A	90	18	CB	EB
1600	7E	D1	43	04	05	CA	44	0A	C5	1E	FF	FE	29	28	08	3E
1610	2C	CD	51	04	CD	1B	17	3E	29	CD	51	04	F1	E3	3D	BE
1620	06	00	30	AA	4F	7E	91	BB	47	38	A3	43	18	A0	E1	3E
1630	28	CD	51	04	CD	95	0E	ED	5B	66	01	D5	3E	2C	CD	51
1640	04	CD	95	0E	ED	5B	66	01	D5	01	FF	00	7E	FE	2C	20
1650	15	C5	CD	18	17	C1	1D	43	1C	28	AA	7E	FE	2C	20	06
1660	C5	CD	18	17	C1	4B	3E	29	CD	51	04	E3	C5	CD	66	15
1670	C1	EB	E1	E3	D5	C5	CD	66	15	C1	D1	78	96	D2	D1	16
1680	ED	44	B9	30	01	4F	23	23	7E	23	66	6F	E5	C5	48	06
1690	00	09	C1	EB	79	96	38	38	3C	4F	46	23	23	7E	23	66



16A0	6F	EB	78	B7	28	1D	C5	06	00	1A	ED	B1	79	C1	20	20
16B0	4F	C5	D5	E5	18	06	13	1A	BE	20	03	23	10	F8	E1	D1
16C0	C1	20	09	D1	ED	52	7D	CD	32	12	E1	C9	79	B7	20	D6
16D0	D1	AF	18	F3	CD	1E	17	4F	ED	78	C3	32	12	CD	0A	17
16E0	ED	79	C9	CD	0A	17	47	C5	1E	00	CD	48	09	28	08	3E
16F0	2C	CD	51	04	CD	1B	17	C1	ED	78	AB	A0	28	FA	C9	EB
1700	3E	29	CD	51	04	C1	D1	C5	43	C9	CD	1B	17	D5	3E	2C
1710	CD	51	04	CD	1B	17	C1	C9	CD	47	09	CD	85	0E	CD	27
1720	0A	7A	B7	C2	44	0A	CD	48	09	7B	C9	CD	9D	15	CA	0E
1730	20	5F	23	23	7E	23	66	6F	E5	19	46	72	E3	C5	7E	CD
1740	DE	23	C1	E1	70	C9	38	09	CD	1B	03	EE	03	4F	C3	1E
1750	03	CD	1B	17	FE	04	D2	6D	04	47	CD	1B	03	E6	FC	B0
1760	18	EB	CD	9F	18	CD	1B	17	FE	0E	DA	44	0A	FD	77	01
1770	4F	D6	0E	30	FC	C6	1C	ED	44	81	FD	77	02	C9	CD	14
1780	0A	DA	6D	04	4F	06	03	CD	0C	03	38	1B	3C	20	F6	10
1790	F6	CD	0C	03	38	11	3C	28	F8	3D	B9	28	05	CD	AB	17
17A0	18	E5	0E	07	C3	0F	03	1E	13	18	11	06	03	CD	B6	17
17B0	B7	20	F8	10	F8	C9	CD	0C	03	D0	1E	14	C3	7E	04	14
17C0	C4	B4	05	18	F5	CD	14	0A	DA	6D	04	4F	CD	47	09	C2
17D0	6D	04	E5	C5	01	FF	08	CD	12	03	10	FB	C1	CD	12	03
17E0	2A	5E	01	EB	2A	5C	01	4E	23	CD	12	03	7C	92	20	02
17F0	7D	93	20	F3	01	FF	08	CD	12	03	10	FB	E1	C9	FE	97

1800	16	FF	28	0B	32	66	01	CD	B4	05	16	00	21	65	01	CD
1810	47	09	CD	7E	17	2A	5C	01	06	03	CD	0C	03	38	A0	5F
1820	96	A2	20	18	73	CD	3E	04	7E	B7	23	20	EB	10	EB	22
1830	5E	01	3A	AB	01	B7	CC	DB	0C	C3	56	05	1E	15	C3	7E
1840	04	C0	E1	01	FF	08	CD	12	03	10	FB	2A	5C	01	7E	23
1850	B6	23	28	31	5E	23	56	23	E5	CD	18	12	CD	DD	24	23
1860	CD	95	18	E1	7E	FE	09	28	05	0E	20	CD	12	03	CD	37
1870	08	E5	21	B1	01	CD	95	18	E1	0E	0D	CD	12	03	0E	0A
1880	CD	12	03	18	C9	0E	1A	CD	12	03	01	FF	08	CD	12	03
1890	10	FB	C3	A0	04	7E	B7	C8	4F	CD	12	03	23	18	F6	EB
18A0	21	FC	05	E3	E5	EB	FD	21	15	03	FD	22	8E	01	FD	21
18B0	95	01	C9	08	EB	2A	54	01	7C	A5	3C	C2	44	0A	2A	5C
18C0	01	7E	23	B6	EB	CA	0D	0B	11	0A	00	D5	08	DC	85	09
18D0	ED	53	A1	01	D1	FE	2C	20	06	CD	47	09	DC	85	09	ED
18E0	53	9F	01	E5	2A	5C	01	23	23	5E	23	56	E1	FE	2C	20
18F0	0E	CD	47	09	DC	85	09	2A	A1	01	ED	52	DA	6D	04	ED
1900	53	AC	01	B7	C2	6D	04	CD	92	05	D2	E1	0A	60	69	D9
1910	11	01	00	21	00	00	D9	11	FF	FF	CD	95	05	D9	2B	EB
1920	ED	4B	9F	01	78	B1	CA	6D	04	CD	C6	23	ED	5B	A1	01
1930	19	DA	6B	11	2A	5C	01	23	23	4E	23	46	EB	2A	AC	01
1940	ED	42	38	06	28	0E	60	69	18	0D	ED	4B	9F	01	2A	9A
1950	01	09	18	03	2A	A1	01	22	9A	01	EB	CD	47	09	B7	CA
1960	68	1A	FE	88	28	14	FE	8C	28	10	FE	A2	28	0C	FE	C9
1970	28	08	FE	9D	28	04	FE	8B	20	E1	CD	47	09	30	DF	22
1980	9C	01	2B	11	00	00	0E	00	CD	47	09	30	2B	E5	B7	21
1990	98	19	ED	52	30	11	21	89	2E	CD	BD	07	CD	9A	1A	E1
19A0	CD	47	09	38	FB	18	B7	62	6B	29	29	19	29	D6	30	5F
19B0	16	00	19	EB	E1	0C	18	D0	79	32	9E	01	E5	2A	AC	01
19C0	EB	E5	ED	52	38	2A	CD	92	05	60	69	D1	D9	2A	A1	01
19D0	ED	5B	9F	01	D9	CD	95	05	38	13	21	75	2E	D5	CD	BD
19E0	07	E1	CD	C9	24	CD	9A	1A	E1	7E	C3	5E	19	D9	E5	D9



19F0	D1	E1	AF	06	98	CD	AB	22	CD	DD	24	06	00	23	E5	7E
1A00	B7	28	04	04	23	18	F8	3A	9E	01	90	28	3C	38	1C	4F
1A10	06	00	2A	9C	01	54	5D	09	E5	2A	5E	01	ED	52	ED	42
1A20	44	4D	E1	ED	B0	ED	53	5E	01	18	1E	ED	44	4F	06	00
1A30	2A	5E	01	54	5D	09	CD	3E	04	22	5E	01	EB	E5	ED	4B
1A40	9C	01	ED	42	44	4D	E1	ED	B8	D1	2A	9C	01	1A	B7	28
1A50	05	77	23	13	18	F7	E5	2A	5C	01	EB	CD	5F	05	E1	7E
1A60	FE	2C	CA	7A	19	C3	5E	19	23	B6	23	B6	2B	C2	37	19
1A70	ED	5B	AC	01	CD	92	05	60	69	ED	4B	9F	01	ED	5B	A1
1A80	01	23	23	73	23	72	23	7E	B7	20	FB	EB	09	EB	23	B6
1A90	23	B6	20	EE	CD	C6	05	C3	A0	04	2A	9A	01	CD	C1	24
1AA0	C3	EE	0C	CD	73	05	D1	C5	CD	92	05	D2	44	0A	54	5D
1AB0	E3	E5	7C	92	20	02	7D	93	D2	44	0A	CD	DB	0C	C1	21
1AC0	56	05	E3	EB	2A	5E	01	1A	02	03	13	7C	92	20	02	7D
1AD0	93	20	F4	ED	43	5E	01	C9	3E	7F	01	3E	BF	F5	CD	9A
1AE0	0E	CD	48	09	20	14	CD	BF	22	3D	2F	47	F1	4F	2F	A0
1AF0	47	3A	A3	01	A1	B0	32	A3	01	C9	F1	C3	6D	04	CD	85
1B00	09	C0	E1	CD	92	05	D2	44	0A	60	69	23	23	4E	23	46
1B10	23	C5	CD	37	08	E1	E5	CD	C9	24	CD	64	07	21	B1	01
1B20	E5	1E	FF	1C	7E	E6	7F	77	23	20	F8	E1	57	06	00	CD
1B30	C9	07	FE	3A	30	10	FE	30	38	0C	D6	30	4F	78	07	07
1B40	80	07	81	47	18	E9	E5	21	2D	1B	E3	05	04	20	01	04
1B50	D9	21	71	1B	BE	23	4E	23	46	23	C5	D9	C8	D9	C1	34
1B60	35	20	F1	FE	60	38	04	E6	5F	18	E6	D9	3E	07	C3	72
1B70	07	20	A4	1B	51	C7	09	4C	DC	1B	46	B5	1B	49	16	1C
1B80	44	E6	1B	0D	65	1C	52	FF	1B	45	68	1C	58	11	1C	4B
1B90	AF	1B	48	0E	1C	7F	59	1C	41	9C	1B	00	C1	D1	CD	EE
1BA0	0C	C3	03	1B	7E	B7	C8	14	CD	66	07	23	10	F6	C9	E5
1BB0	21	FA	1B	E3	37	F5	CD	C9	07	4F	F1	35	34	C8	F5	DC
1BC0	FA	1B	7E	CD	66	07	F1	F5	30	05	CD	75	1C	18	02	23
1BD0	14	7E	B7	28	05	B9	20	EB	10	E9	F1	C9	CD	2A	08	CD
1BE0	EE	0C	C1	C3	15	1B	7E	B7	C8	3E	5C	CD	72	07	7E	B7
1BF0	28	08	CD	66	07	CD	75	1C	10	F4	3E	5C	C3	72	07	7E

1C00	B7	C8	CD	C9	07	CD	66	07	77	23	14	10	F2	C9	36	00
1C10	5A	06	FF	CD	A4	1B	CD	C9	07	FE	0D	28	48	FE	1B	C8
1C20	FE	7F	20	0F	15	14	28	12	2B	15	7E	CD	66	07	CD	75
1C30	1C	18	E3	F5	7B	FE	FF	38	08	F1	3E	07	CD	72	07	18
1C40	D5	92	1C	14	D5	EB	6F	26	00	19	44	4D	03	CD	1F	04
1C50	D1	F1	CD	66	07	77	23	18	BD	7A	B7	C8	15	2B	7E	CD
1C60	66	07	10	F5	C9	CD	2A	08	CD	EE	0C	C1	D1	37	F5	21
1C70	B1	01	C3	F9	04	E5	1D	7E	B7	28	07	23	7E	2B	77	23
1C80	18	F5	E1	C9	C0	EB	21	87	01	ED	5F	77	23	77	23	77
1C90	23	77	23	E6	7F	77	23	36	80	EB	C9	CD	9F	18	C0	E5
1CA0	2A	5E	01	CD	EE	0C	CD	93	09	ED	5B	60	01	7C	92	20
1CB0	02	7D	93	28	3B	4E	23	7E	23	CB	7F	20	2B	CD	72	07
1CC0	79	E6	7F	C4	72	07	3E	24	CB	79	C4	72	07	3E	3D	CD
1CD0	72	07	79	17	9F	22	66	01	E5	20	09	CD	EC	22	CD	DD
1CE0	24	CD	ED	13	CD	34	14	E1	23	23	23	23	23	23	18	B3
1CF0	E1	C9	CD	B4	05	2A	5E	01	2B	CD	64	1D	38	27	28	F9
1D00	FE	7F	28	F5	23	77	CD	3E	04	7E	D6	0D	20	13	77	23
1D10	77	23	77	23	77	CD	64	1D	38	0B	FE	0A	28	DB	B7	18
1D20	DD	FE	0D	20	D4	36	00	23	36	1A	2A	5E	01	E5	7E	D6
1D30	1A	20	09	32	AB	01	CD	C2	05	C3	A0	04	7E	23	B7	20



1D40	FB	7E	23	B7	28	FB	2B	22	9A	01	E1	3E	FF	32	AB	01
1D50	CD	48	09	3C	3D	CA	A9	04	30	05	F5	AF	C3	F1	04	1E
1D60	18	C3	7E	04	CD	0C	03	D8	E6	7F	C9	CD	B4	05	21	B0
1D70	01	06	00	CD	64	1D	38	32	28	F9	FE	7F	28	F5	FE	1A
1D80	28	28	FE	0A	20	04	04	05	28	E9	4F	78	FE	FF	79	28
1D90	02	23	04	77	D6	0D	20	DB	77	3E	FE	32	AB	01	21	B0
1DA0	01	CD	47	09	3C	3D	28	C6	18	AE	CD	C2	05	C3	A0	04
1DB0	32	66	01	CD	47	09	11	FF	FF	28	0E	3E	2C	CD	51	04
1DC0	D2	6D	04	CD	85	09	C2	6D	04	ED	53	9A	01	3E	FD	32
1DD0	AB	01	C3	07	18	3C	CA	2D	1D	3C	28	92	CD	C2	05	AF
1DE0	32	AB	01	ED	5B	9A	01	7A	A3	3C	01	CC	08	C5	C8	C3
1DF0	DA	0A	C8	D2	44	0A	CD	85	09	ED	53	9A	01	11	0A	00
1E00	FE	2C	20	06	CD	47	09	DC	85	09	ED	53	9C	01	3E	AD
1E10	CD	51	04	CD	73	05	D1	C5	60	69	D9	11	01	00	21	00
1E20	00	D9	CD	95	05	D2	44	0A	E5	D9	EB	ED	4B	9C	01	78
1E30	B1	CA	6D	04	CD	C6	23	ED	5B	9A	01	19	DA	6B	11	E5
1E40	ED	5B	9A	01	CD	92	05	DA	44	0A	D1	C5	60	69	7E	23
1E50	B6	28	0A	23	7E	23	66	6F	ED	52	DA	6B	11	C1	E1	D1
1E60	E5	ED	52	E3	2B	ED	42	EB	38	0A	ED	42	DA	44	0A	09
1E70	EB	E1	E5	19	E3	E5	50	59	ED	4B	5E	01	09	E5	CD	19
1E80	04	E1	22	5E	01	60	69	C1	E3	D5	ED	B0	E1	ED	4B	9A
1E90	01	23	23	71	23	70	23	7E	B7	20	FB	23	D1	7C	92	20
1EA0	02	7D	93	28	0B	D5	EB	2A	9C	01	09	44	4D	EB	18	E1
1EB0	CD	DB	0C	C3	56	05	CD	6A	10	D5	E5	21	6D	01	CD	19
1EC0	23	2A	60	01	E3	3A	02	01	F5	3E	2C	CD	51	04	CD	6A
1ED0	10	C1	3A	02	01	A8	1F	DA	90	0E	E3	EB	E5	2A	60	01
1EE0	7C	92	20	02	7D	93	C2	44	0A	D1	E1	E3	D5	CD	19	23
1EF0	E1	11	6D	01	CD	19	23	E1	C9	3E	01	32	4E	01	C3	6A
1F00	10	E5	19	EB	2A	62	01	B7	ED	52	E3	C1	EB	1B	1B	1B
1F10	1B	28	02	ED	B0	ED	53	62	01	32	4E	01	E1	7E	FE	2C
1F20	C0	CD	47	09	18	D3	C8	ED	73	9A	01	E5	11	57	1F	D5
1F30	CD	24	0A	0E	00	D5	CD	48	09	28	10	3E	2C	CD	51	04
1F40	0C	C5	CD	24	0A	C1	EB	E3	EB	18	EA	D5	EB	2A	9A	01
1F50	2B	72	2B	73	2B	2B	C9	E1	C9	3E	00	C4	1B	17	C0	FE
1F60	0B	28	F6	D2	44	0A	32	AF	01	C9	CD	27	0A	1A	C3	32
1F70	12	CD	24	0A	D5	3E	2C	CD	51	04	CD	1B	17	D1	12	C9
1F80	21	08	2A	CD	05	23	18	0C	CD	05	23	18	04	C1	DD	E1
1F90	D1	CD	D2	22	78	B7	C8	3A	6B	01	B7	CA	F5	22	90	30
1FA0	11	ED	44	D9	DD	E5	CD	02	23	D9	DD	E3	CD	F5	22	D9
1FB0	DD	E1	FE	29	D0	F5	CD	21	23	67	F1	CD	8F	20	B4	21
1FC0	66	01	F2	D6	1F	CD	5A	20	30	69	23	34	CA	55	20	2E
1FD0	01	CD	AC	20	18	5D	AF	90	47	7E	9B	5F	23	7E	9A	57
1FE0	23	7E	DD	9D	DD	6F	23	7E	DD	9C	DD	67	23	7E	99	4F
1FF0	DC	72	20	68	63	AF	47	79	B7	20	27	DD	4C	DD	7D	DD

2000	67	DD	6A	AF	54	65	6F	78	D6	08	FE	D0	20	E8	AF	32
2010	6B	01	C9	05	29	CB	12	08	DD	29	08	30	02	DD	23	08
2020	CB	11	F2	13	20	78	5C	45	B7	28	08	21	6B	01	86	77
2030	30	DC	C8	78	21	6B	01	B7	FC	45	20	46	23	7E	E6	80
2040	A9	4F	C3	F5	22	1C	C0	14	C0	DD	2C	C0	DD	24	C0	0C
2050	C0	0E	80	34	C0	1E	06	C3	7E	04	7E	83	5F	23	7E	8A
2060	57	23	7E	DD	8D	DD	6F	23	7E	DD	8C	DD	67	23	7E	89
2070	4F	C9	21	6C	01	7E	2F	77	AF	6F	67	90	47	7D	ED	52
2080	EB	6F	DD	9D	DD	6F	7D	DD	9C	DD	67	7D	99	4F	C9	06



2090	00	D6	08	38	10	43	5A	DD	55	08	DD	7C	DD	6F	08	DD
20A0	61	0E	00	18	EC	C6	09	6F	AF	2D	C8	79	1F	4F	DD	7C
20B0	1F	DD	67	DD	7D	1F	DD	6F	CB	1A	CB	1B	CB	18	18	E8
20C0	00	00	00	00	00	81	06	23	85	AC	C3	11	7F	53	CB	9E
20D0	B7	23	7F	CC	FE	A6	0D	53	7F	CB	5C	60	BB	13	80	DD
20E0	E3	4E	38	76	80	5C	29	3B	AA	38	82	CD	BF	22	B7	EA
20F0	44	0A	21	6B	01	7E	01	35	80	DD	21	F3	04	11	FA	33
2100	90	F5	70	D5	DD	E5	C5	CD	94	1F	C1	DD	E1	D1	04	CD
2110	C3	21	21	C0	20	CD	88	1F	21	C6	20	CD	DB	2A	01	80
2120	80	DD	21	00	00	11	00	00	CD	94	1F	F1	CD	A1	24	01
2130	31	80	DD	21	17	72	11	D2	F7	18	04	C1	DD	E1	D1	CD
2140	BF	22	C8	2E	00	CD	63	22	79	D5	D9	4F	D1	DD	E5	E1
2150	D9	01	00	00	50	58	DD	21	00	00	21	F3	1F	E5	21	68
2160	21	E5	E5	E5	E5	21	66	01	7E	23	B7	20	0E	43	5A	DD
2170	55	08	DD	7C	DD	6F	08	DD	61	4F	C9	E5	EB	1E	08	1F
2180	57	79	30	12	E5	D9	E3	19	E3	EB	DD	E5	E3	ED	5A	E3
2190	DD	E1	EB	89	D9	E1	1F	4F	DD	7C	1F	DD	67	DD	7D	1F
21AD	DD	6F	CB	1C	CB	1D	CB	18	1D	7A	20	D3	EB	E1	C9	CD
21B0	DA	22	01	20	84	DD	21	00	00	11	00	00	CD	F5	22	C1
21C0	DD	E1	D1	CD	BF	22	CA	70	04	2E	FF	CD	63	22	FD	E5
21D0	34	34	2B	E5	D9	E1	4E	2B	56	2B	5E	2B	7E	2B	6E	67
21E0	EB	D9	41	EB	DD	E5	FD	E1	AF	4F	57	5F	DD	21	00	00
21F0	32	AE	01	E5	FD	E5	C5	E5	78	D9	E3	B7	ED	52	E3	EB
2200	FD	E5	E3	ED	52	E3	FD	E1	EB	99	D9	E1	47	3A	AE	01
2210	DE	00	3F	30	09	32	AE	01	F1	F1	F1	37	18	04	C1	FD
2220	E1	E1	0C	0D	1F	FA	5E	22	17	CB	13	CB	12	08	DD	29
2230	08	30	02	DD	23	08	CB	11	29	08	FD	29	08	30	02	FD
2240	23	08	CB	10	3A	AE	01	17	32	AE	01	79	B2	B3	DD	B4
2250	DD	B5	20	9F	E5	21	6B	01	35	E1	20	97	18	2D	FD	E1
2260	C3	34	20	78	B7	28	20	7D	21	6B	01	AE	80	47	1F	A8
2270	78	F2	86	22	C6	80	77	CA	AD	21	CD	21	23	77	2B	C9
2280	CD	BF	22	2F	B7	21	B7	E1	F2	0E	20	C3	55	20	CD	02
2290	23	78	B7	C8	C6	02	38	F3	47	CD	94	1F	21	6B	01	34
22A0	C0	18	E8	CD	BF	22	06	88	11	00	00	21	6B	01	4F	D5
22B0	DD	E1	11	00	00	70	06	00	23	36	80	17	C3	F0	1F	3A
22C0	6B	01	B7	C8	3A	6A	01	FE	2F	17	9F	C0	3C	C9	CD	BF
22D0	22	F0	21	6A	01	7E	EE	80	77	C9	EB	2A	66	01	E3	E5
22E0	2A	68	01	E3	E5	2A	6A	01	E3	E5	EB	C9	11	66	01	01
22F0	06	00	ED	B0	C9	ED	53	66	01	DD	22	68	01	ED	43	6A
2300	01	C9	21	66	01	5E	23	56	23	4E	DD	69	23	4E	DD	61
2310	23	4E	23	46	23	C9	11	66	01	01	06	00	EB	ED	B0	EB
2320	C9	21	6A	01	7E	07	37	1F	77	3F	1F	23	23	77	79	07
2330	37	1F	4F	1F	AE	C9	78	B7	28	85	CD	BF	22	79	28	88
2340	21	6A	01	AE	79	FA	C8	22	CD	50	23	1F	A9	C3	C8	22
2350	23	78	BE	C0	2B	79	BE	C0	2B	DD	7C	BE	C0	2B	DD	7D
2360	BE	C0	2B	7A	BE	C0	2B	7B	96	C0	E1	C9	5E	23	56	23
2370	4E	23	46	23	C9	47	4F	57	5F	DD	67	DD	6F	B7	C8	E5
2380	CD	02	23	CD	21	23	AE	67	F2	9B	23	1B	7A	A3	3C	20
2390	0A	DD	2B	DD	7C	DD	A5	3C	20	01	0D	3E	A8	90	CD	8F
23A0	20	7C	17	DC	45	20	06	00	DC	72	20	E1	C9	21	6B	01
23B0	7E	FE	A8	3A	66	01	D0	7E	CD	75	23	36	A8	7B	F5	79
23C0	17	CD	F0	1F	F1	C9	21	00	00	78	B1	C8	3E	11	3D	C8
23D0	29	38	08	EB	29	EB	30	F6	09	30	F3	C3	6B	11	FE	26
23E0	CA	71	24	FE	2D	F5	28	05	FE	2B	28	01	2B	CD	0E	20
23F0	47	57	5F	2F	4F	CD	47	09	38	56	FE	2E	28	26	FE	45



2400	20	25	CD	47	09	15	FE	A6	28	0E	FE	2D	28	0A	14	FE
2410	2B	28	05	FE	A5	28	01	2B	CD	47	09	38	49	14	20	07
2420	AF	93	5F	0C	0C	28	CE	E5	7B	90	F2	39	24	CD	DA	22
2430	ED	44	21	C0	20	CD	EC	22	37	F5	CD	AE	24	20	FB	F1
2440	30	07	C1	DD	E1	D1	CD	C3	21	D1	F1	CC	D2	22	EB	C9
2450	D5	57	78	89	47	C5	E5	D5	CD	8E	22	F1	D6	30	CD	A1
2460	24	E1	C1	D1	18	8F	7B	87	87	83	87	86	D6	30	5F	18
2470	A7	11	00	00	CD	47	09	28	22	38	0C	D6	41	38	1C	FE
2480	06	30	18	C6	0A	18	02	D6	30	08	7A	FE	10	D2	55	20
2490	08	EB	29	29	29	29	B5	6F	EB	18	D9	E5	CD	18	12	E1
24A0	C9	CD	DA	22	CD	A6	22	C1	DD	E1	D1	C3	94	1F	C8	F5
24B0	CD	8E	22	F1	3D	C9	D5	E5	F5	CD	AF	21	F1	E1	D1	3C
24C0	C9	E5	21	93	2F	CD	BD	07	E1	CD	17	12	21	AF	01	7E
24D0	36	00	F5	CD	DD	24	F1	32	AF	01	C3	30	14	AF	32	AA
24E0	01	21	6E	01	36	20	E6	08	28	02	36	2B	CD	BF	22	F2
24F0	FA	24	36	2D	E5	CD	D2	22	E1	B4	23	36	30	3A	AA	01
2500	57	17	DA	B2	25	CA	AA	25	E5	CD	1F	27	21	AF	01	34
2510	35	28	51	57	C6	0B	FA	55	25	BE	28	02	30	37	47	7E
2520	90	3C	4F	04	7A	16	0B	E1	23	CD	83	26	E5	AF	01	C0
2530	00	ED	B1	2B	01	3C	25	C5	AF	F5	18	47	E1	7E	FE	2D
2540	C8	FE	20	C8	FE	30	28	0A	FE	25	20	05	23	7E	FE	2D
2550	C8	2B	36	20	C9	4E	0D	28	01	0C	06	02	E1	23	7A	16
2560	00	C3	D7	26	01	00	03	C6	0C	FA	74	25	FE	0D	30	04
2570	3C	47	3E	02	D6	02	E1	F5	CD	5A	27	36	30	20	01	23
2580	CD	6D	27	2B	7E	FE	30	28	FA	FE	2E	28	01	23	F1	28
2590	1A	36	45	23	36	2B	F2	9D	25	36	2D	ED	44	06	2F	04
25A0	D6	0A	30	FB	C6	3A	23	70	23	77	23	36	00	EB	21	6E
25B0	01	C9	23	C5	E5	7A	1F	DA	CE	26	01	0E	B6	DD	21	C9
25C0	1B	11	04	BF	CD	36	23	FA	D3	25	E1	C1	CD	DD	24	2B
25D0	36	25	C9	16	0B	CD	BF	22	C4	1F	27	E1	C1	FA	83	26
25E0	C5	5F	78	92	93	F4	F9	27	CD	01	28	CD	6D	27	B3	C4
25F0	1B	28	B3	C4	5A	27	D1	7B	B7	20	01	2B	3D	F4	F9	27
2600	E5	21	6E	01	46	0E	20	3A	AA	01	5F	E6	20	28	07	78
2610	B9	0E	2A	20	01	41	71	CD	47	09	28	10	FE	45	28	0C
2620	FE	30	28	F2	FE	2C	28	EE	FE	2E	20	03	2B	36	30	CB
2630	63	28	03	2B	36	24	CB	53	20	02	2B	70	E1	28	02	70
2640	23	36	00	21	6D	01	23	3A	A8	01	95	92	C8	7E	FE	20
2650	28	F4	FE	2A	28	F0	2B	E5	F5	CD	47	09	FE	2D	28	F8
2660	FE	2B	28	F4	FE	24	28	F0	FE	30	20	10	23	CD	47	09
2670	30	0A	2B	01	2B	77	F1	28	FB	C1	18	CB	F1	28	FD	E1
2680	36	25	C9	5F	79	B7	28	01	3D	83	FA	8E	26	AF	C5	F5
2690	FC	B6	24	20	FB	C1	7B	90	C1	5F	82	78	FA	AA	26	92
26A0	93	F4	F9	27	C5	CD	01	28	18	11	CD	F9	27	79	CD	5D
26B0	27	4F	AF	92	93	CD	F9	27	C5	47	4F	CD	6D	27	C1	B1
26C0	20	03	2A	A8	01	83	3D	F4	F9	27	50	C3	00	26	CD	BF
26D0	22	37	C4	1F	27	E1	C1	F5	79	B7	F5	28	01	3D	80	4F
26E0	7A	E6	04	FE	01	9F	57	81	4F	D6	0B	F5	C5	FC	B6	24
26F0	FA	ED	26	C1	F1	C5	F5	FA	FB	26	AF	ED	44	80	3C	82
2700	47	0E	00	CD	6D	27	F1	F4	15	28	C1	F1	20	01	2B	F1
2710	38	04	C6	0B	90	92	C5	CD	91	25	EB	D1	C3	00	26	D5
2720	AF	F5	CD	48	27	01	15	A2	DD	21	F8	02	11	FD	FF	CD
2730	36	23	F2	45	27	F1	CD	AF	24	F5	C3	25	27	F1	CD	B6
2740	24	F5	CD	48	27	F1	D1	C9	01	3A	A5	DD	21	B7	43	11



2750	FC	3F	CD	36	23	E1	F2	3D	27	E9	05	20	08	36	2E	22
2760	A8	01	23	48	C9	0D	C0	36	2C	23	0E	03	C9	D5	C5	E5
2770	CD	80	1F	3C	CD	75	23	CD	F5	22	E1	C1	11	C2	27	3E
2780	0B	CD	5A	27	C5	F5	E5	D5	CD	02	23	E1	06	2F	04	7B
2790	96	5F	23	7A	9E	57	23	DD	7D	9E	DD	6F	23	DD	7C	9E
27A0	DD	67	23	79	9E	4F	2B	2B	2B	2B	30	E2	CD	5A	20	23
27B0	CD	F5	22	EB	E1	70	23	F1	C1	3D	20	C5	CD	5A	27	77
27C0	D1	C9	00	E4	0B	54	02	00	CA	9A	3B	00	00	E1	F5	05
27D0	00	80	96	98	00	00	40	42	0F	00	00	A0	86	01	00	00
27E0	10	27	00	00	00	E8	03	00	00	00	64	00	00	00	00	0A
27F0	00	00	00	00	01	00	00	00	00	B7	C8	3D	36	30	23	18

2800	FB	7B	82	3C	47	3C	D6	03	30	FC	C6	05	4F	3A	AA	01
2810	E6	40	C0	4F	C9	20	04	C8	CD	5A	27	36	30	23	3D	18
2820	F6	CD	47	09	30	1E	CD	85	09	E5	CD	92	05	D2	E1	0A
2830	60	69	23	23	23	CD	47	09	D6	9C	C2	44	0A	47	CD	F0
2840	13	E1	18	03	CD	95	0E	CD	48	09	37	28	0C	FE	2C	28
2850	05	FE	3B	C2	6D	04	CD	47	09	EB	2A	66	01	01	D1	EB
2860	E5	F5	D5	46	B0	CA	44	0A	23	23	4E	23	66	69	C3	77
2870	28	CD	F9	29	CD	72	07	AF	5F	57	CD	F9	29	57	7E	23
2880	FE	23	CA	BE	28	FE	27	CA	81	29	05	CA	6D	29	FE	2B
2890	3E	08	28	E6	2B	7E	23	FE	2E	CA	D6	28	BE	20	D2	FE
28A0	24	28	14	FE	2A	20	CA	78	FE	02	23	38	03	7E	FE	24
28B0	3E	20	20	07	05	1C	FE	AF	C6	10	23	1C	82	57	1C	0E
28C0	00	05	28	46	7E	23	FE	2E	28	17	FE	23	28	F0	FE	2C
28D0	20	19	CB	F2	18	E8	7E	FE	23	3E	2E	C2	71	28	0E	01
28E0	23	0C	05	28	25	7E	23	FE	23	28	F6	D5	54	5D	FE	5E
28F0	20	16	BE	20	13	23	BE	20	0F	23	BE	20	0B	23	78	D6
2900	04	38	05	D1	47	14	23	CA	EB	D1	2B	1C	CB	5A	20	13
2910	1D	78	B7	28	0E	7E	D6	2D	28	06	FE	FE	20	05	CB	DA
2920	CB	D2	05	E1	F1	28	51	C5	D5	CD	9A	0E	D1	C1	C5	E5
2930	43	78	81	FE	19	D2	44	0A	7A	F6	80	CD	DE	24	CD	31
2940	14	E1	CD	48	09	37	28	0C	FE	3B	28	05	FE	2C	C2	6D
2950	04	CD	47	09	C1	EB	E1	E5	F5	D5	7E	90	23	23	4E	23
2960	66	69	16	00	5F	19	78	B7	C2	77	28	18	06	CD	F9	29
2970	CD	72	07	E1	F1	C2	5E	28	DC	EE	0C	E3	CD	66	15	E1
2980	C9	0E	01	1E	4C	05	28	1C	7E	23	FE	45	28	0C	FE	52
2990	28	08	FE	4C	28	04	FE	43	20	0A	5F	0C	05	28	05	7E
29A0	23	BB	28	F7	CD	F9	29	E1	F1	28	CD	C5	D5	CD	95	0E
29B0	D1	C1	C5	E5	2A	66	01	41	0E	00	7B	FE	45	28	2B	D5
29C0	C5	CD	CD	15	C1	D1	78	96	47	7B	FE	4C	28	0E	FE	52
29D0	28	15	78	CB	38	90	08	CD	F1	29	08	47	C5	CD	34	14
29E0	C1	CD	F1	29	C3	41	29	AF	18	EC	78	96	30	ED	AF	18
29F0	EA	04	05	C8	CD	64	07	18	F9	F5	7A	B7	3E	2B	C4	72
2A00	07	F1	C9	21	D2	22	E3	E9	00	00	00	00	00	80	CD	DA
2A10	22	21	08	2A	CD	EC	22	C1	DD	E1	D1	CD	BF	22	28	44
2A20	78	B7	CA	0F	20	D5	DD	E5	C5	79	F6	7F	CD	02	23	F2
2A30	44	2A	D5	DD	E5	C5	CD	AD	23	C1	DD	E1	D1	F5	CD	36
2A40	23	E1	7C	1F	E1	22	6A	01	E1	22	68	01	E1	22	66	01
2A50	DC	03	2A	CC	D2	22	D5	DD	E5	C5	CD	EB	20	C1	DD	E1
2A60	D1	CD	3F	21	01	38	81	DD	21	3B	AA	11	5C	29	CD	3F
2A70	21	3A	6B	01	FE	88	D2	80	22	CD	DA	22	CD	AD	23	C1
2A80	DD	E1	D1	F5	CD	91	1F	21	9E	2A	CD	EA	2A	21	6B	01
2A90	F1	B7	FA	97	2A	86	01	86	3F	77	D0	C3	80	22	0A	CC



2AA0	D5	45	56	15	6A	CF	37	A0	92	27	6D	F5	95	EE	93	00
2AB0	71	D0	FC	A7	78	21	74	B1	21	82	C4	2E	77	82	58	58
2AC0	95	1D	7A	6D	CB	46	58	63	7C	E9	FB	EF	FD	75	7E	D2
2AD0	F7	17	72	31	80	00	00	00	00	00	81	CD	DA	22	11	3B
2AE0	21	D5	E5	CD	02	23	CD	3F	21	E1	CD	DA	22	7E	23	CD
2AF0	EC	22	FE	F1	C1	DD	E1	D1	3D	C8	D5	DD	E5	C5	F5	E5
2B00	CD	3F	21	E1	CD	05	23	E5	CD	94	1F	E1	18	E5	CD	BF
2B10	22	FA	35	2B	21	87	01	CD	EC	22	C8	01	35	98	DD	21
2B20	7A	44	11	00	00	CD	3F	21	01	28	68	DD	21	46	B1	11
2B30	00	00	CD	94	1F	CD	02	23	7B	59	4F	36	80	2B	46	36
2B40	80	CD	F3	1F	21	87	01	C3	16	23	21	9C	2B	CD	83	1F
2B50	CD	DA	22	01	49	83	DD	21	DA	0F	11	21	A2	CD	F5	22
2B60	C1	DD	E1	D1	CD	C3	21	CD	DA	22	CD	AD	23	C1	DD	E1
2B70	D1	CD	91	1F	21	A2	2B	CD	88	1F	CD	BF	22	37	F2	88
2B80	2B	CD	80	1F	CD	BF	22	B7	F5	F4	D2	22	21	A2	2B	CD
2B90	83	1F	F1	D4	D2	22	21	A8	2B	C3	DB	2A	21	A2	DA	0F
2BA0	49	81	00	00	00	00	00	7F	07	90	BA	34	76	6A	82	E4
2BB0	E9	E7	4B	F1	84	B1	4F	7F	3B	28	86	31	B6	64	69	99
2BC0	87	E4	36	E3	35	23	87	24	31	E7	5D	A5	86	21	A2	DA
2BD0	0F	49	83	CD	DA	22	CD	50	2B	C1	DD	E1	D1	CD	DB	22
2BE0	EB	CD	F5	22	CD	4A	2B	C3	BF	21	CD	BF	22	FC	03	2A
2BF0	FC	D2	22	3A	6B	01	FE	81	38	10	01	00	81	DD	21	00

2C00	00	51	59	CD	C3	21	21	88	1F	E5	21	14	2C	CD	DB	2A
2C10	21	9C	2B	C9	0D	14	07	BA	FE	62	75	51	16	CE	D8	D6
2C20	78	4C	BD	7D	D1	3E	7A	01	CB	23	C4	D7	7B	DC	3A	0A
2C30	17	34	7C	36	C1	A3	81	F7	7C	EB	16	61	AE	19	7D	5D
2C40	78	8F	60	B9	7D	A2	44	12	72	63	7D	16	62	FB	47	92
2C50	7E	C0	F0	BF	CC	4C	7E	7E	8E	AA	AA	AA	7F	F6	FF	FF
2C60	FF	7F	80	45	4E	C4	46	4F	D2	4E	45	58	D4	44	41	54
2C70	C1	49	4E	50	55	D4	44	49	CD	52	45	41	C4	4C	45	D4
2C80	47	4F	20	54	CF	46	4E	45	4E	C4	49	C6	52	45	53	54
2C90	4F	52	C5	47	4F	20	53	55	C2	52	45	54	55	52	CE	52
2CA0	45	CD	53	54	4F	D0	4F	55	D4	4F	CE	4E	55	4C	CC	57
2CB0	41	49	D4	44	45	C6	50	4F	4B	C5	50	52	49	4E	D4	BF
2CC0	4C	49	53	54	45	CE	43	4C	45	41	D2	46	4E	52	45	54
2CD0	55	52	CE	53	41	56	C5	A1	55	53	49	4E	C7	54	41	42
2CE0	A8	54	CF	46	CE	53	50	43	A8	54	48	45	CE	4E	4F	D4
2CF0	53	54	45	D0	AB	AD	AA	AF	DE	41	4E	C4	4F	D2	BE	BD
2D00	BC	53	47	CE	49	4E	D4	41	42	D3	55	53	D2	46	52	C5
2D10	49	4E	D0	50	4F	D3	53	51	D2	52	4E	C4	4C	4F	C7	45
2D20	58	D0	43	4F	D3	53	49	CE	54	41	CE	41	54	CE	50	45
2D30	45	CB	4C	45	CE	53	54	52	A4	56	41	CC	41	53	C3	43
2D40	48	52	A4	4C	45	46	54	A4	52	49	47	48	54	A4	4D	49
2D50	44	A4	4C	50	4F	D3	49	4E	53	54	D2	45	4C	53	C5	4C
2D60	50	52	49	4E	D4	54	52	41	43	C5	4C	54	52	41	43	C5
2D70	52	41	4E	44	4F	4D	49	5A	C5	53	57	49	54	43	C8	4C
2D80	57	49	44	54	C8	4C	4E	55	4C	CC	57	49	44	54	C8	4C
2D90	56	41	D2	4C	4C	56	41	D2	53	50	45	41	CB	A7	50	52
2DA0	45	43	49	53	49	4F	CE	43	41	4C	CC	4B	49	4C	CC	45
2DB0	58	43	48	41	4E	47	C5	4C	49	4E	C5	4C	4F	41	44	47
2DC0	CF	52	55	CE	4C	4F	41	C4	4E	45	D7	41	55	54	CF	43
2DD0	4F	50	D9	41	4C	4F	41	44	C3	41	4D	45	52	47	45	C3
2DE0	41	4C	4F	41	C4	41	4D	45	52	47	C5	41	53	41	56	C5



2DF0	4C	49	53	D4	4C	4C	49	53	D4	52	45	4E	55	4D	42	45
2E00	D2	44	45	4C	45	54	C5	45	44	49	D4	43	4F	4E	D4	00
2E10	7C	53	4E	45	58	54	20	57	2F	4F	20	46	4F	D2	53	59
2E20	4E	54	41	58	20	45	52	52	4F	D2	52	45	54	55	52	4E
2E30	20	57	2F	4F	20	47	4F	53	55	C2	4F	55	54	20	4F	46
2E40	20	44	41	54	C1	49	4C	4C	45	47	41	4C	20	46	55	4E
2E50	43	54	49	4F	CE	41	52	49	54	48	4D	45	54	49	43	20
2E60	4F	56	45	52	46	4C	4F	D7	4F	55	54	20	4F	46	20	4D
2E70	45	4D	4F	52	D9	55	4E	44	45	46	49	4E	45	44	20	53
2E80	54	41	54	45	4D	45	4E	54	A0	53	55	42	53	43	52	49
2E90	50	54	20	4F	55	54	20	4F	46	20	52	41	4E	47	C5	52
2EA0	45	2D	44	49	4D	45	4E	53	49	4F	4E	45	44	20	41	52
2EB0	52	41	D9	43	41	4E	27	54	20	2F	B0	49	4C	4C	45	47
2EC0	41	4C	20	44	49	52	45	43	D4	54	59	50	45	20	4D	49
2ED0	53	2D	4D	41	54	43	C8	4E	4F	20	53	54	52	49	4E	47
2EE0	20	53	50	41	43	C5	53	54	52	49	4E	47	20	54	4F	4F
2EF0	20	4C	4F	4E	C7	54	4F	4F	20	43	4F	4D	50	4C	45	D8
2F00	43	41	4E	27	54	20	43	4F	4E	54	49	4E	55	C5	55	4E
2F10	44	45	46	49	4E	45	44	20	55	53	45	52	20	43	41	4C
2F20	CC	46	49	4C	45	20	4E	4F	54	20	46	4F	55	4E	C4	49
2F30	4C	4C	45	47	41	4C	20	45	4F	C6	46	49	4C	45	53	20
2F40	44	49	46	46	45	52	45	4E	D4	52	45	43	4F	56	45	52
2F50	45	C4	46	4E	52	45	54	55	52	4E	20	57	2F	4F	20	46
2F60	55	4E	43	54	49	4F	4E	20	43	41	4C	CC	4D	49	53	53
2F70	49	4E	47	20	53	54	41	54	45	4D	45	4E	54	20	4E	55
2F80	4D	42	45	D2	2A	49	4E	56	41	4C	49	44	20	49	4E	50
2F90	55	54	8A	20	40	20	4C	49	4E	45	A0	0A	52	45	41	44
2FA0	59	3A	8A	2A	45	58	54	52	41	20	4C	4F	53	54	8A	0A
2FB0	2A	42	52	45	41	CB	00	CD	EE	0C	2A	10	2E	CD	C9	24
2FC0	CD	EE	0C	21	B5	30	CD	BD	07	AF	21	00	01	3F	77	2C
2FD0	20	FC	24	38	F8	2B	F9	22	5A	01	21	FF	FF	22	54	01
2FE0	3E	2C	32	B0	01	3E	C3	32	00	00	32	8D	01	32	A4	01
2FF0	21	C9	2F	22	A5	01	FD	21	90	01	21	0F	03	22	8E	01

3000	3E	48	32	91	01	32	96	01	3E	38	32	92	01	32	97	01
3010	21	03	03	22	01	00	21	08	01	22	06	01	21	00	80	22
3020	87	01	22	89	01	22	8B	01	21	EF	30	CD	BD	07	CD	E2
3030	06	CD	47	09	FE	41	28	8B	2F	FE	AC	CA	B7	2F	3C	20
3040	07	CD	21	03	60	6F	18	14	21	B1	01	CD	48	09	CD	DE
3050	23	7E	B7	C2	6D	04	CD	27	0A	EB	2B	2B	22	04	01	22
3060	48	01	E5	11	00	03	B7	ED	52	E1	DA	4D	04	11	9C	FF
3070	19	11	00	03	7C	92	20	02	7D	93	11	00	34	30	03	11
3080	00	03	7C	92	20	02	7D	93	38	E0	F9	22	5A	01	EB	22
3090	5C	01	CD	3E	04	B7	EB	ED	52	01	F0	FF	09	CD	EE	0C
30A0	CD	C9	24	21	FE	30	CD	BD	07	21	BD	07	22	A5	01	CD
30B0	B4	05	C3	A0	04	54	2E	44	2E	4C	2E	20	5A	2D	38	30
30C0	20	42	41	53	49	43	20	62	79	20	4E	65	69	6C	20	43
30D0	6F	6C	76	69	6E	20	26	20	52	6F	67	65	72	20	41	6D
30E0	69	64	6F	6E	0A	4D	61	79	20	20	31	39	37	37	8A	0A
30F0	48	69	67	68	65	73	74	20	4D	65	6D	6F	72	F9	20	42
3100	79	74	65	73	20	46	72	65	65	0A	0A	57	65	6C	63	6F
3110	6D	65	20	74	6F	20	42	41	53	49	43	2C	20	56	65	72
3120	2E	20	32	2E	31	0A	3C	54	44	4C	20	5A	2D	38	30	20
3130	48	69	67	68	20	50	72	65	63	69	73	69	6F	6E	20	45



3140	78	74	65	6E	64	65	64	20	56	65	72	73	69	6F	6E	3E
3150	8A	00	00	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3160	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3170	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00F	FF	00	FF	00
3180	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3190	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
31A0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
31B0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
31C0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	0
31D0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
31E0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
31F0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3200	9F	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3210	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3220	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3230	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3240	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3250	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3260	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3270	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3280	9F	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3290	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
32A0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
32B0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
32C0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
32D0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
32E0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3300	DF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3310	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3320	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3330	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3340	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3350	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3360	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3370	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
3380	FF	00	FF	00	FF0	00	FF	00	FF	00	FF	00	FF	00	FF	00
3390	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
33A0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
33B0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
33C0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
33D0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
33E0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00
33F0	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00



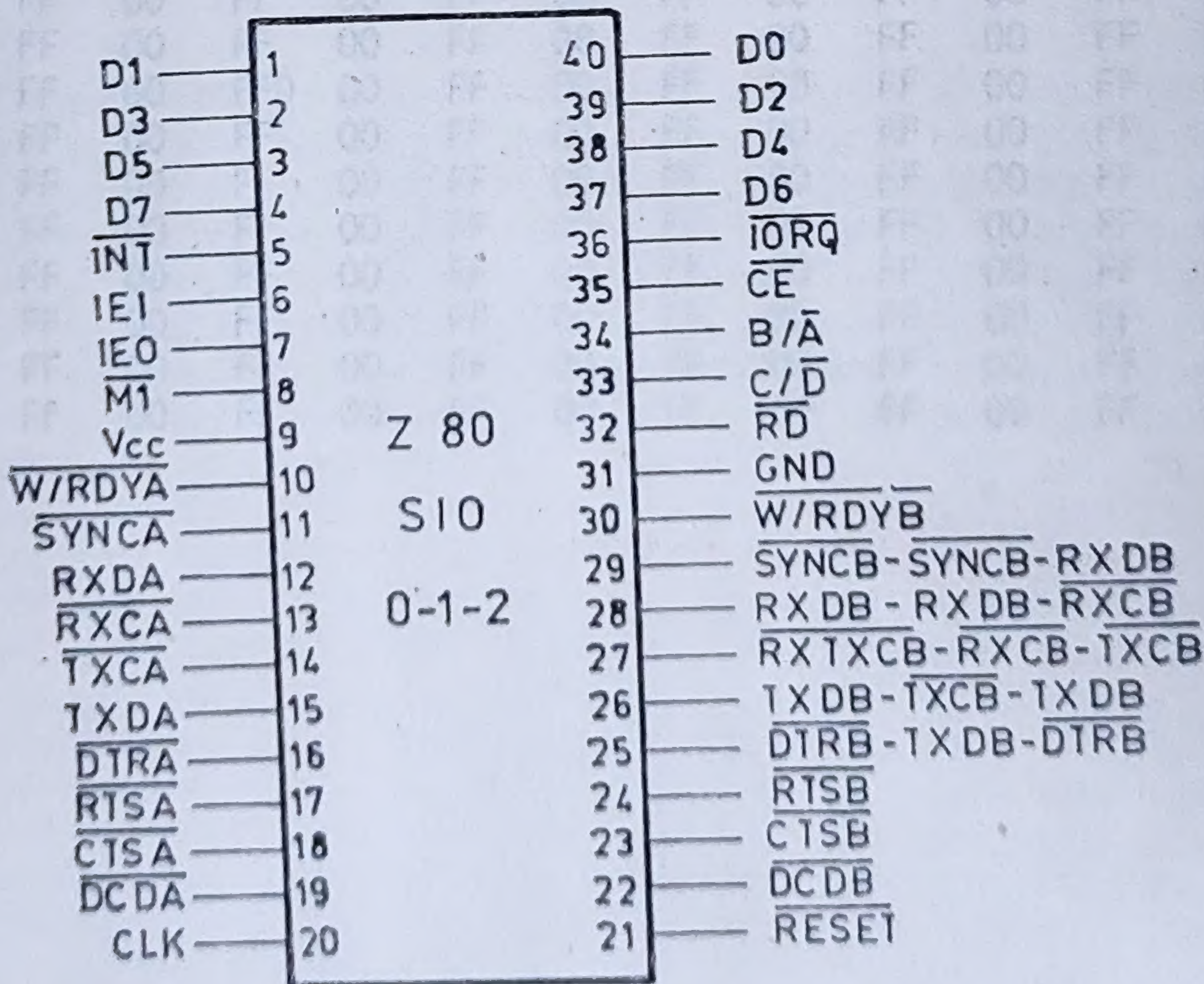
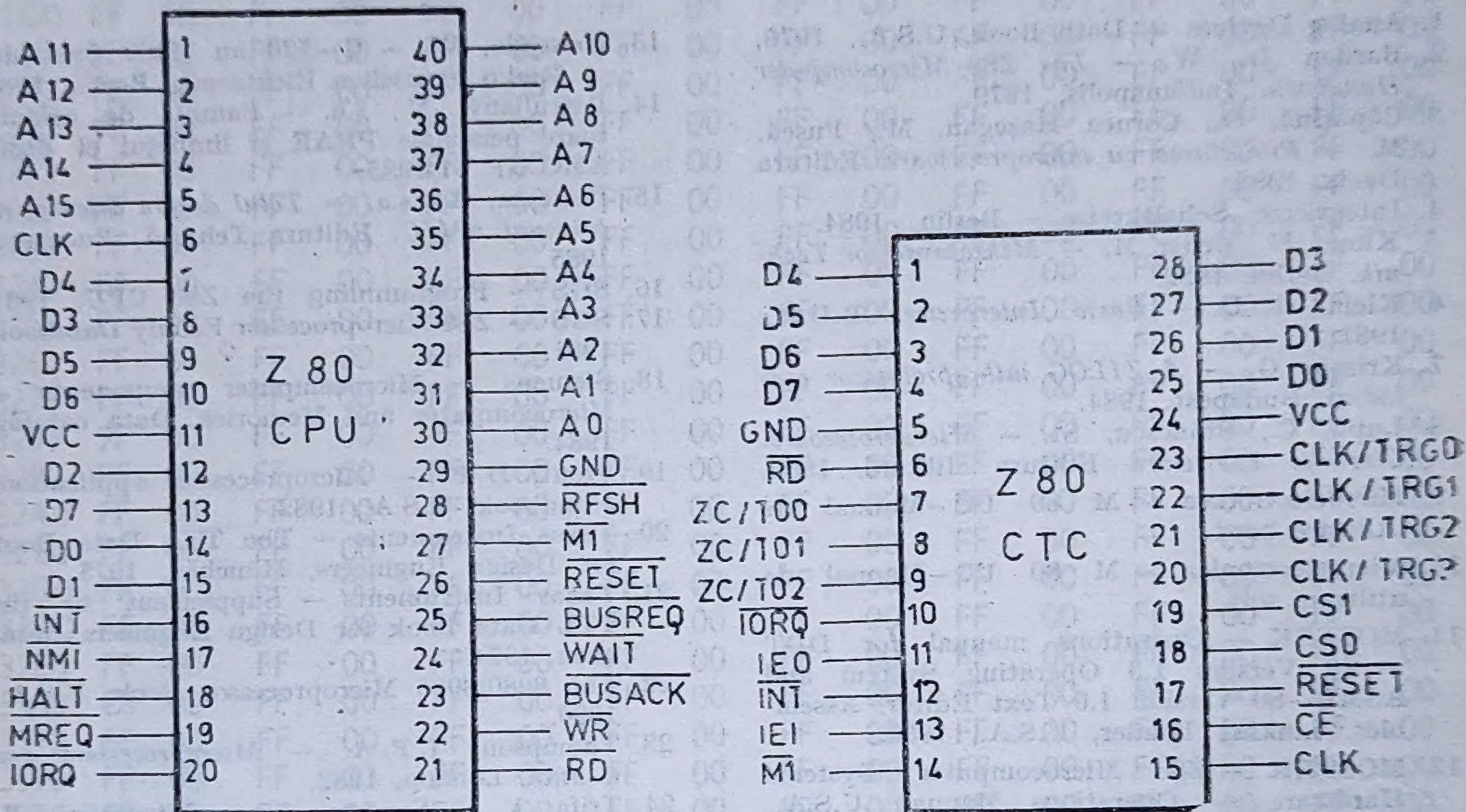
## BIBLIOGRAFIA

1. Analog Devices — Data Book, U.S.A., 1976.
2. Barden Jr., W. — *The Z80 Microcomputer Handbook*. Indianapolis, 1979.
3. Căpățină, O., Cornea Hașegan, M., Pușcă, M. — *Proiectarea cu microprocesoare*. Editura Dacia, 1983.
4. Integrierte Schaltkreise — Berlin, 1984.
5. Kieser, H., Meder, M. — *Mikroprozessor Technik*. Berlin, 1982.
6. Klein, R. D. — *Basic Interpreter*. B.D.R., 1981.
7. Krizsán G. — *A ZILOG mikroprocesszor családjai*. Budapest, 1984.
8. Lupu, C., Stănescu, St. — *Microprocesoare. Circuite. Proiectare*. Editura Militară, 1986.
9. Microelectronica — M 80 UC—Manual de utilizare hard.
10. Microelectronica — M 80 UC—Manual de utilizare soft.
11. MOSTEK — Operations manual for DDT — 80 Version 1.3 Operating System and ASMB—80 Version 1.0 Text Editor, Assembler, Linking Loader, U.S.A., 1980.
12. MOSTEK — Z80 Microcomputer Systems Hardware — Operations Manual, U.S.A., 1980.
13. Namyslo, W. — C-520 an U880 Systemen — Radio Fernsehen Elektronik, Berlin, 1985.
14. Patrubby, N., ș.a. — Familia de calculatoare personale PRAE și limbajul ei Basic, AMC Nr. 51/1985.
15. Petrescu, A., ș.a. — *Totul despre calculatorul personal aMIC*. Editura Tehnică, București, 1985.
16. SGS — Programming the Z80 CPU, 1981.
17. SGS — Z80 Microprocessor Family Databook, 1981.
18. Siemens — Microcomputer components — Microcomputer and Memories, Data catalog, 1984.
19. Stout, D. F. — Microprocessor Applications Handbook, U.S.A., 1982.
20. Texas Instruments — The TTL Data Book for Design Engineers, München, 1975.
21. Texas Instruments — Supplement to the TTL Data Book for Design Engineers, München, 1974.
22. The 8080/8085 Microprocessor Book, U.S.A., 1980.
23. Thompson, J. F. A. — *Microprocessors and Control*. London, 1982.
24. Trifa, V. — *Servomecanisme*. Litografia I.P. C-N, 1981.

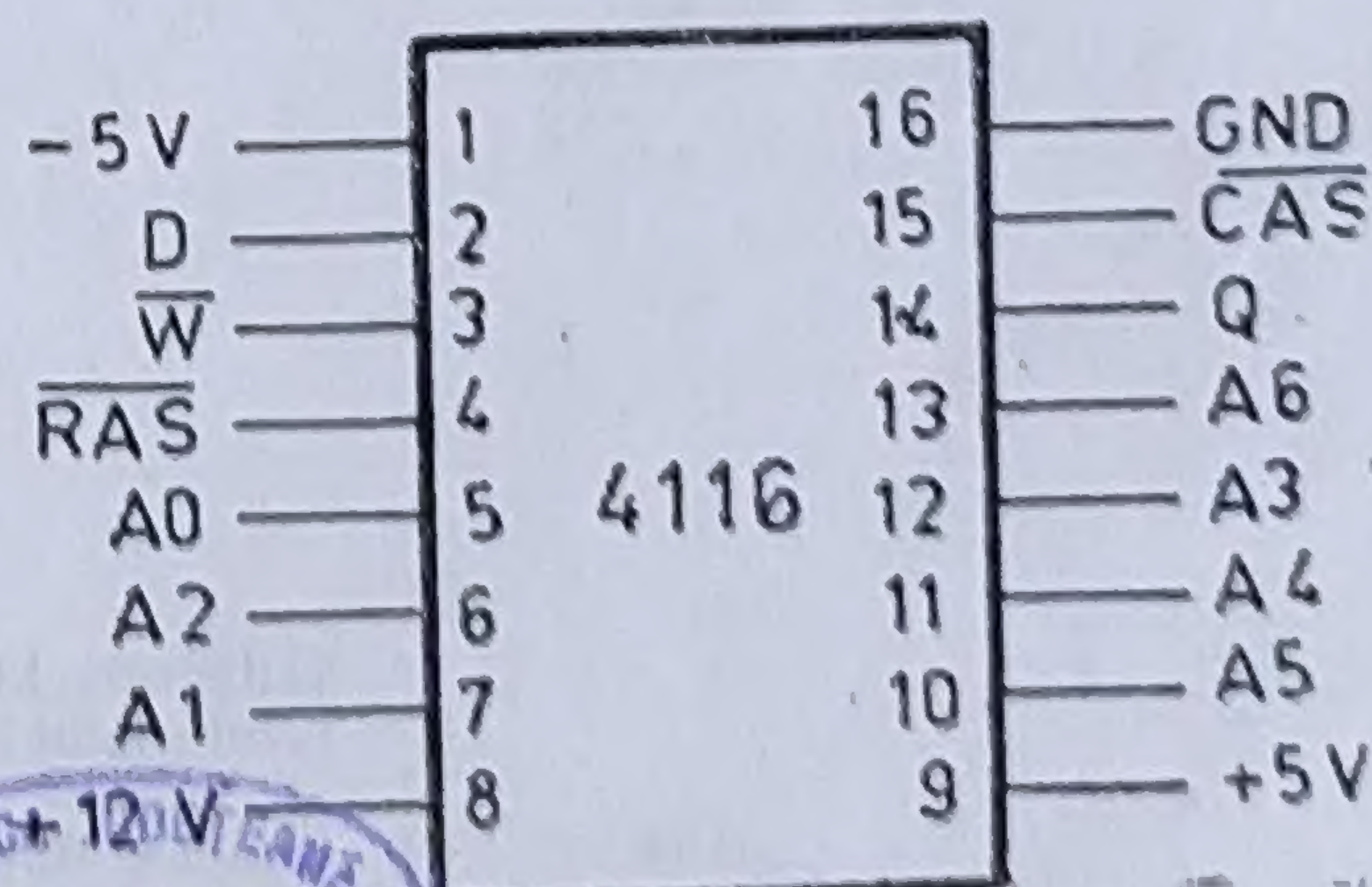
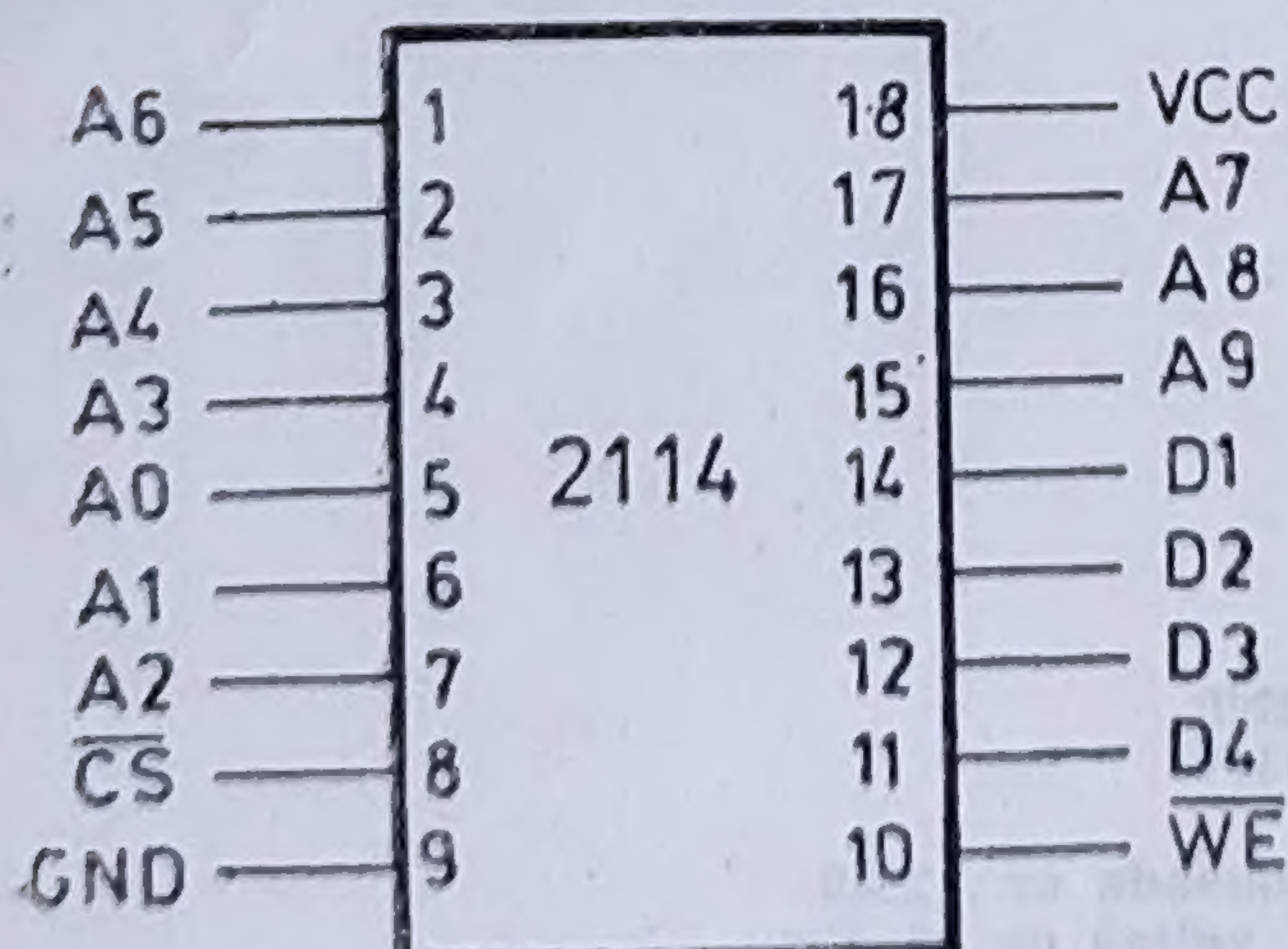
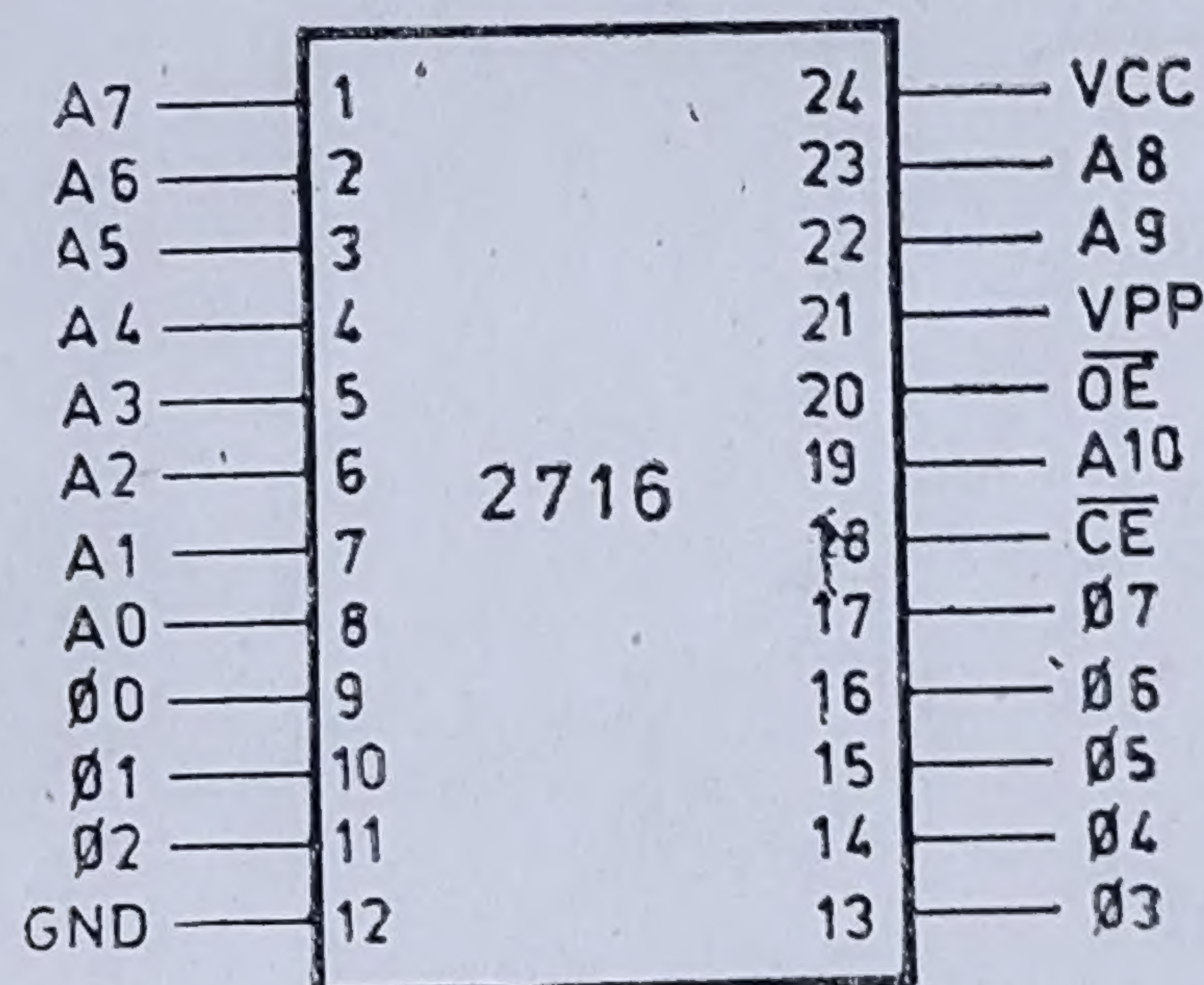
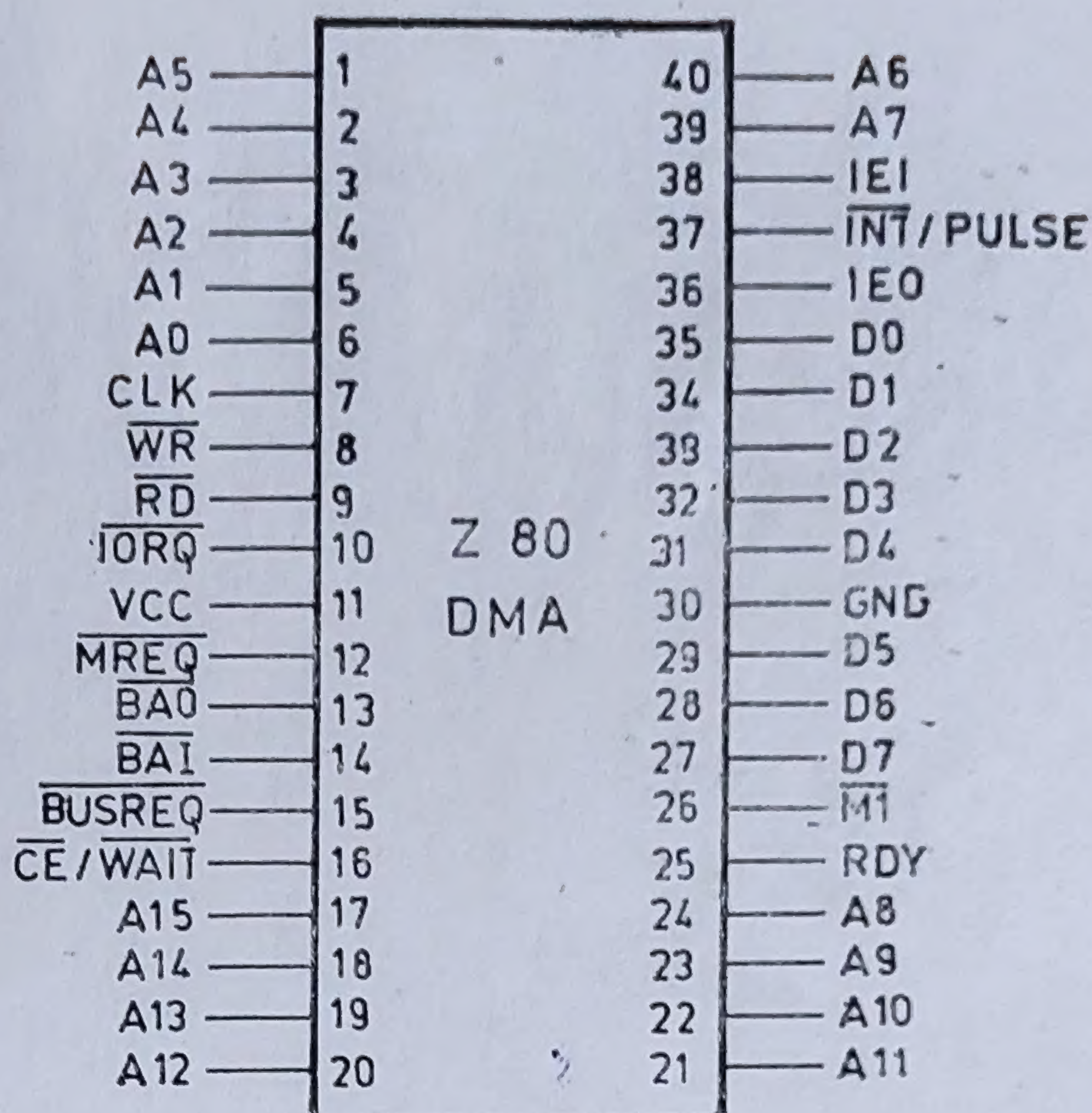
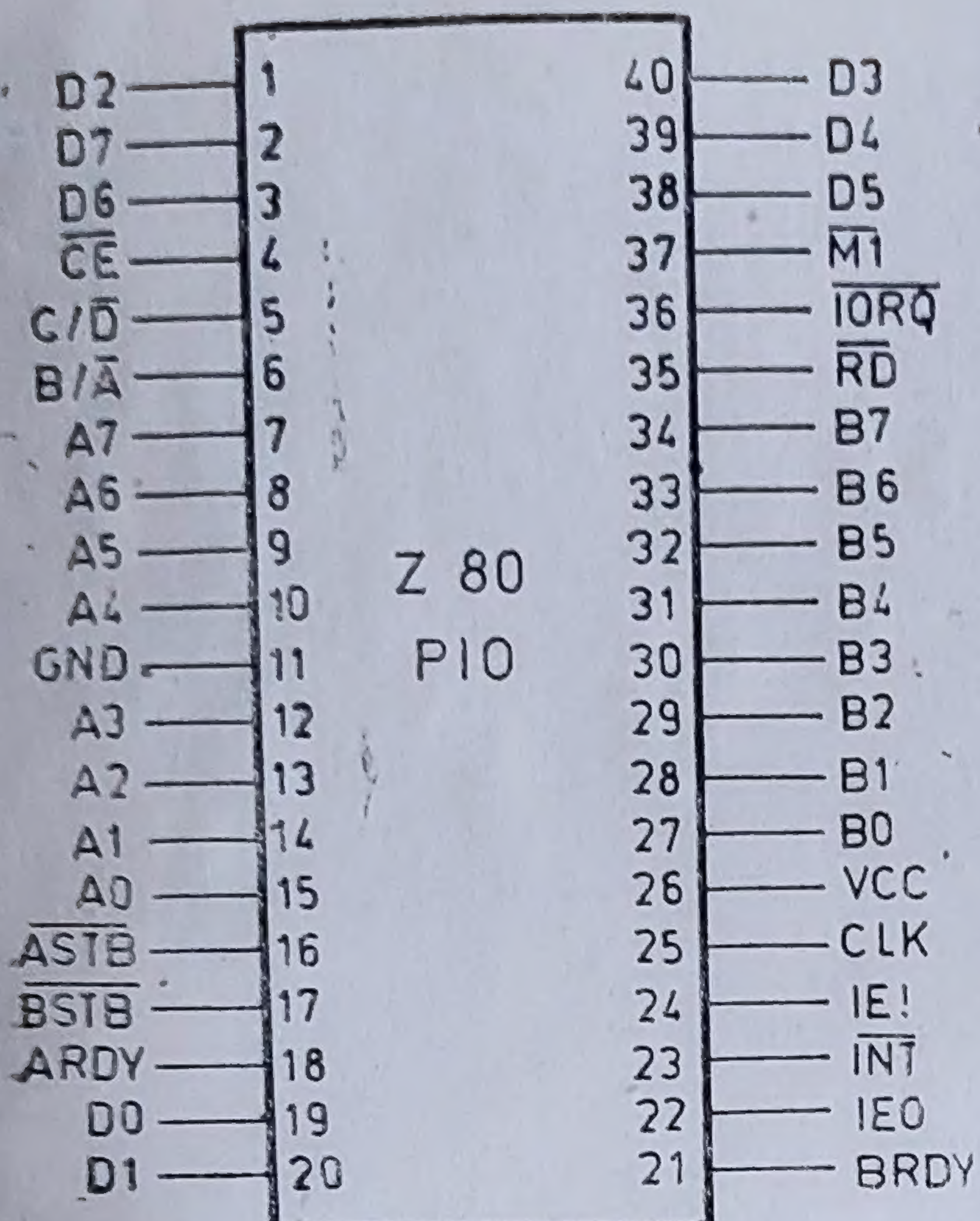


# ANEXA

Conexiunile principalelor circuite utilizate în sistemele cu microprocesor Z80 sunt prezentate în continuare.







BIBLIOTEC + 12 MEANS  
OH ASAGHI  
IASI



Lei 23, -

